2

A071692

INTELLIGENT BANDWIDTH COMPRESSION

D.Y. Tseng, B.L. Bullock, K.E. Olin, R.K. Kandt

Hughes Research Laboratories
3011 Malibu Canyon Road
Malibu, CA 90265

J.D. Olsen

Ground Systems Group
Hughes Aircraft Company
Fullerton, CA 92634

February 1980

DAAK70–78–C–0148
FINAL REPORT

Approved for public release, distribution unlimited

[redacted]

Sponsored By
NIGHT VISION LABORATORY
Fort Belvoir, Virginia 22060

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A085 284

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

(1) INTELLIGENT BANDWIDTH COMPRESSION,

5. TYPE OF REPORT & PERIOD COVERED

(9) Final Rep[redacted]t, 1 Nov 78-31 Jul 79,

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

(10) D.Y./Tseng, B.L./Bullock, K.E./Olin
R.K./Kandt, J.D./Olsen

8. CONTRACT OR GRANT NUMBER(s)

(15) DAAK70-78-C-0148

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Hughes Research Laboratories
3011 Malibu Canyon Road
Malibu, California 90265

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

(12) 114

11. CONTROLLING OFFICE NAME AND ADDRESS

Night Vision Laboratories
Fort Belvoir, Virginia 22060

12. REPORT DATE

(11) Feb [redacted] 80

13. NUMBER OF PAGES

122

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS (of this report)

UNCLASSIFIED

15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image compression
Bandwidth compression
Scene-analysis
Knowledge-based system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The feasibility of a 1000:1 bandwidth compression ratio for image transmission has been demonstrated using image-analysis algorithms and a rule-based controller. Such a high compression ratio was achieved by first analyzing scene content using auto-cueing and feature-extraction algorithms, and then transmitting only the pertinent information consistent with mission requirements. A rule-based controller directs the flow of analysis and performs priority allocations on the extracted

172600 SLB

scene content. The reconstructed bandwidth-compressed image consists of an edge map of the scene background, with primary and secondary target windows embedded in the edge map. The bandwidth-compressed images are updated at a basic rate of 1 frame per second, with the high-priority target window updated at 7.5 frames per second. The scene-analysis algorithms used in this system together with the adaptive priority controller are described. Results of simulated 1000:1 band-width-compressed images are presented. A video tape simulation of the Intelligent Bandiwdth Compression system has been produced using a sequence of video input from the data base.

TABLE OF CONTENTS

A

LIST OF ILLUSTRATIONS

PRECEDING PAGE BLANK-NOT FI

FIGURE PAGE

SECTION 1

INTRODUCTION AND SUMMARY

This is the final technical report for the Intelligent Bandwidth Compression (IBC) Program (Contract No. DAAK70-78-C-0148). The report describes the results of a nine-month technical effort to develop techniques and algorithms for achieving a 1,000:1 bandwidth compression in image data transmission rate for remotely piloted vehicle (RPV) applications. The period covered by this report is 1 November 1978 to 31 July 1979. Mr. David Singer is the contract monitor.

The overall objective of this program is to develop high-ratio bandwidth-compression techniques based on image understanding, advanced scene analysis, and spatial and temporal image sampling. Our approach to achieving a 1,000:1 bandwidth compression of RPV derived imagery utilizes the information-extraction and image-understanding techniques developed in the Army ATAC and DARPA Terminal Homing programs in conjunction with a knowledge-based control scheme developed under a Hughes IR&D program. Scene-analysis techniques are used to extract key features and detect targets in the images. Knowledge-based control schemes are applied to adaptively evaluate and match the available targets with mission goals for priority assignments and to make decisions on the allocation of resources in the available channel bandwidth capacity. Conventional coding techniques are used to encode the resources for further compression for transmission to, and reconstruction by, the ground station.

PRECEDING PAGE BLANK-NOT FILMED

Several salitent aspects of this program are summarized below:

- Demonstrated feasibility of 1,000:1 compression ratio in image transmission for RPV applications
- Performed IBC simulation study on all images in NVL-supplied data base.
- Utilized a top-down design procedure in software development.
- Implemented a knowledge-based system in PASCAL language.
- Incorporated scene analysis components such as auto-cuer, feature-extraction, model building as the foundation to the IBC system.
- Tested a model-based classified for auto-cuer.
- Produced a video tape simulation of the IBC system using a sequence from the NVL video data base.

The IBC system consists of three major components: an image interpreter, an adaptive priority controller, and a data line encoder/decoder. Figure 1-1 shows these major components and their interactive connections. The image interpreter operates on digitized images from the IBC data base and performs three primary operations: target detection, statistical target identification, and structural/feature analysis. Control information about where (i.e., addresses in the input image) and how (i.e., specifications of features and analysis procedures to be used) the interpretation is to be performed is supplied by the adaptive priority controller. This particular control path is the most important in the system because it closes the control loop on the interpretation system, allowing interactive refinement of the results. The image interpreter provides a list of the target descriptors that go to the adaptive priority controller and a collection of image descriptors that go to the data link encoder/decoder.

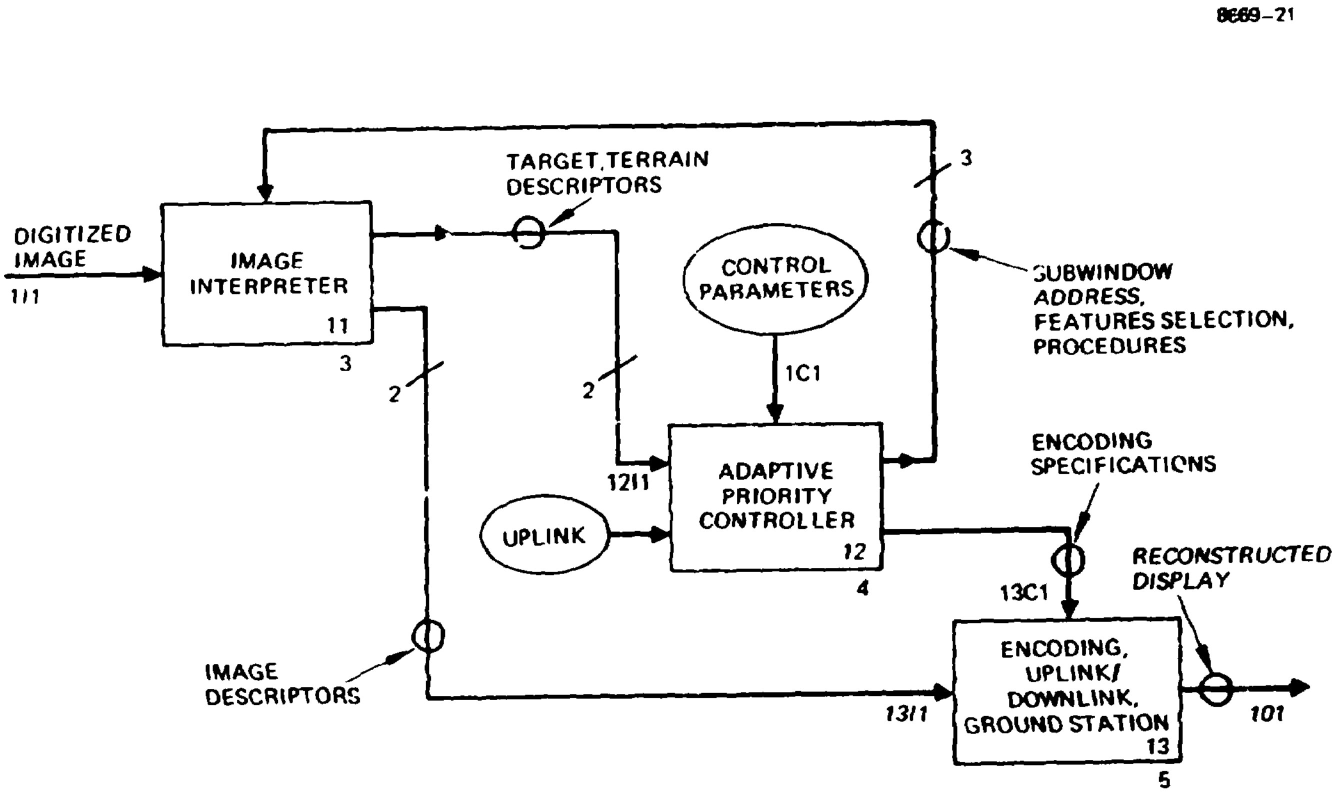


Figure 1-1. IBC system components.

Three types of target descriptors are contained in the list sent to the adaptive priority controller: a cued target at a given location, an initial classification of the target, and a structurally refined scene-analysis classification. The adaptive priority controller utilizes these inputs to determine what type of processing is necessary and what the image interpreter should be doing next. The control policy for this decision is contained in a table of control parameters that lists the target and scene object priorities as well as the allocations allowed for different scene componeits for the encoded signals to be transmitted. Through the uplink input, the operator can change this table of control parameters dynamically to suit his needs.

The image descriptors from the image interpreter consist of an edge image target subwindows, and target/object descriptors. These are encoded by the encoder/decoder, transmitted through a stimulated down-link (with jamming and noise), and decoded for reconstruction in the simulated ground station for display. This display represents a 1,000:1 bandwidth compressed image of the original sensed scene.

The resource allocation for available channel capacity in the 1,000:1 bandwidth compression scheme is shown in Table 1-1. The channel capacity requirements used for image-compression comparison is based on an image data rate derived from a 512 x 512 pixel, 8-bit, 30 frames-per-sec (fps) fideo data stream. Under this assumption, the channel capacity required for full rate image transmission is 63 Mbit/sec. Thus, for the 1,000:1 bandwidth compression required, the available channel capacity for image transmission is reduced to 63 kbit/sec. As shown in Table 1-1, the IBC program produces the required 1,000:1 compression while maintaining all essential scene information by allocating resources as follows:

- A 32 x 32 pixel high-priority target window transmitted at full resolution and updated at 7.5 fps.
- An orientation (or reference) window of 128 x 128 pixel size centered about the detected (high-priority) targets and updated at 1 fps.
- An edge map of the image reduced in resolution by a factor of 2 and replicated to 256 x 256 pixel size at the ground station. The update rate is also 1 fps.
- Six secondary target windows of 32 x 32 pixel size updated at 1 fps.
- Approximately 200 symbolic descriptors to denote scene content information of interest, both supplementing and complementing the above stated resources.

Table 1-1. IBC Resource Allocation

Data Composition	Pixels	fps	BPP	Data Rate kbit/sec
Edge map	256 x 256[a]	1.0	1.0	16.4
Priority target window	32 x 32	7.5	1.5	11.5
Orientation window	128 x 128	1.0	1.0	16.4
Secondary target windows(6)	32 x 32	1.0	1.5	9.2
Symbolic Descriptors	——	1.0	——	9.5
			Total	63.0

[a] Edge map transmitted at reduced resolution (128 x 128 pixel) and replicated to 256 x 256 size at ground station.

The exact composition of the resource allocation can by dynamically changed, depending on the scene content. This can be accomplished either automatically by the adaptive priority controller or manually by the operator through the uplink command. For example, the secondary targets and a portion of the symbolic descriptor allocations can (if desired) be redirected to provide an additional high-priority target window at a 7.5 fps update rate.

Details of the IBC system, together with examples of the reconstructed images, are presented in this report. A system overview, which discusses the motivation for using this approach, together with the software system implementation methodology and a top-down design of the IBC system, is included in Section 2. The image interpreter component of the IBC system, comprised of the ATAC auto-cuer and an edge feature-extractor, is discussed in Section 3. The knowledge-based controller, the heart of the IBC system, is presented in Section 4. That section also includes the actual production rules used in the IBC system. Section 5 contains the results of conventional coding of the resources processed by the IBC system. These resources are used in the reconstruction of the 1,000:1 bandwidth compressed image. Examples of reconstructed images derived from the simulation studies are also shown in Section 5. Conclusions and recommendations are discussed in Section 6.

SECTION 2

SYSTEM OVERVIEW

A. BASIC APPROACH

Our approach to achieving a 1,000:1 bandwidth compression of RPV-derived imagery is based on image-understanding and information-extraction techniques developed in the Army ATAC and DARPA Terminal Homing programs, in conjunction with a knowledge-based control scheme developed under an IR&D program. Scene-analysis techniques are used for extracting key features and detecting targets in the images. Knowledge-based control schemes are applied to adaptively evaluate and match the available targets with mission goals for priority assignments and to make decisions on the allocation of resources in the available channel bandwidth capacity. Conventional coding techniques are used to encode the resources for transmission to, and reconstruction by, the ground station. The system that has been constructed is especially significant because it is a first attempt at doing intelligent coding using scene-analysis and knowledge-based system techniques, it is our first attempt at constructing a PASCAL (i.e., non-LISP) knowledge-based system, and it is our first effort at applying modest software-engineering techniques to a research (e.g., nonproduction) software construction. The overall system strategy can be summarized as follows:

- Concentrate first on finding likely interest areas using ATAC-like cueing.
- Refine this classification by using advanced, model-driven techniques to extract target detail, target group

formations, context information, and terrain cues (roads, tracks, etc.).

- Refine the resulting symbolic description of the scene by iterating these steps to extract new detail.
- Transmit a compressed and encoded version of the scene.

A block diagram of the system, as originally conceived in the IBC proposal, is given in Figure 2-1. The functional blocks include the sensor and analog-to-digital (A/D) conversion hardware, followed by the frame buffer; both operate at the full frame rate. Image data are accessed directly from the frame memory for subimage video encoding and for the feature-extraction and segmentation functions. Depending on the sensor mode, the feature-extraction and segmentation buffer will hold a single frame (search-mode) or an average of several motion-compensated frames (track mode).

The feature-extraction and segmentation functions include low-level image-processing operations of edge extraction and filtering, statistical and texture-based region segmentation, and local statistical interest operators for target cueing. These operations generate edge element groups for curvilinear line fitting, and statistical data for region-growing and interest-area definitions.

There follows three parallel, higher-level feature-analysis functions: the low-level interest window detectors, the high-level target-cueing classifier, and the model feature detector. On the first pass through these feature-analysis functions, only the low-level interest area detector is used. This detector uses a linear discriminant made up

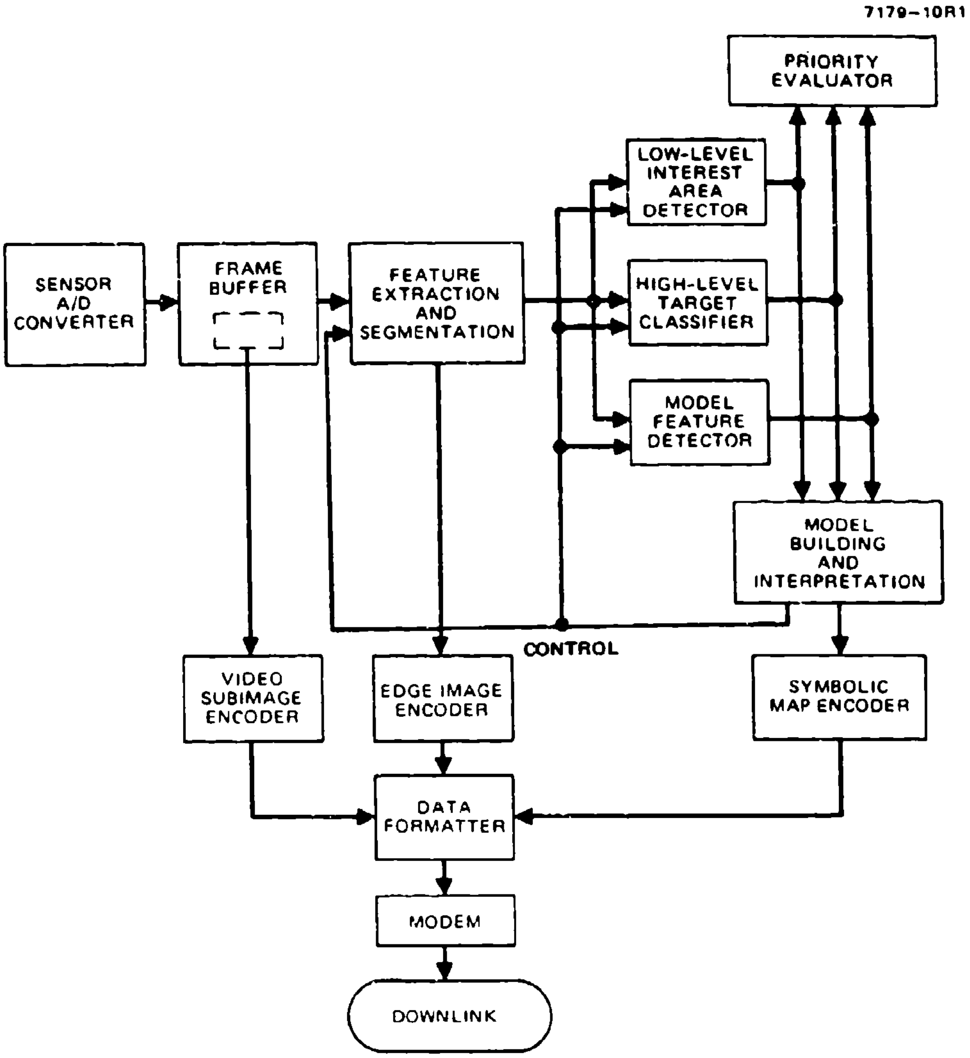


Figure 2-1. Simplified functional block diagram of the IBC system.

of subarea statistical parameters selected to cue high-interest areas in the image. These interest areas are cycled through the remaining two feature-analysis functions to detect and classify targets in the high-interest areas and to derive scene context in and around the high-interest areas, particularly those containing identified targets and target complexes.

The high-level target classifier segments the interest areas using ATAC-like target feature extraction and classification techniques and statistical consistency within segmented regions to aid in identifying candidate targets. The model feature detector builds high-level features, all of which can contribute to the scene-interpretation process.

In the priority evaluator, interest areas are queued and priorities set for the target classification and the model feature detector. The outputs of these feature-analysis functions are also queued and prioritized for the model-building and interpretation functions. Target models are generated and classified on the basis of their positions in this queue as well as a selection of what scene content is to appear in the symbolic scene map.

The model-building and interpretation function controls what will be done next and what information will appear in the encoded image. This is accomplished with a simple knowledge-based production system in which the partial results are matched against a set of prestored control and processing rules. When a match is found, an associated action description determines the next sequence of processing steps to be executed.

The remaining tasks preparatory to data transmission to the ground station are (1) symbolic and edge-element-data encoding and (2) the data formatting function (which collects the encoded data blocks, and, based on data-block priorities, dynamically allocates the available data rate and bandwidth). Conventional coding and bandwidth-reduction techniques are used in the three encoders to further reduce the required bandwidth for the image data blocks. The size and generation rate of the symbolic scene model varies with target density and scene complexity. Similarly, the number of subimages being transmitted at or near the full frame rate will vary with automatic and operator-keyed control. Thus, the bandwidth allocation problem to meet the constantly changing number, type, and priorities of encoded data blocks takes on added significance. The important function ot the data formatter is the dynamic allocation of data rate within the available bandwidth to an optimum set of prioritized data blocks.

B. SOFTWARE SYSTEM IMPLEMENTATION METHODOLOGY

This section describes a methodology for software system implementation now under development at HRL. The IBC system described in this report was our first attempt at using this methodology, and the results were very encouraging.

Software is often delivered late and is frequently extremely unreliable and difficult to maintain. The IBM OS project, for example, required over 5000 man-years of effort and still was late. Henceforth, HRL will use current design and programming techniques, which are embodied in a discipline known as software engineering. Structured analysis and design techniques (SADT) are applied during the design phase of the project, while structured programming techniques in an appropriate language are emphasized during the coding process.

SADT is a coherent methodology for analysis and design, it includes a graphic language for building models, a methodology for developing models from system requirements, and management practices for controlling development. The resulting model is an organized sequence of diagrams starting from a high-level overview of the whole subject. Each component of the overview is then detailed on another diagram, which is further decomposed until enough detail is shown. A graphical representation of this structured decomposition is shown in Figure 2-2. Each diagram shows a limited amount of detail in easy-to-grasp units. Each step of the analysis or design introduces a small amount of further detail about a well-bounded subject, which can be completely contained on one page. This division from less detailed to more detailed is called a top-down design. By dividing the design into structured units, a bounded context is formed at any level of detail. Whenever a diagram is studied, presented, or implemented, it is done in this bounded context so that it is easy to determine what is included and excluded.

The resulting SADT diagram model formed in this manner can then be used for documentation, as a guide in implementation, and for project management. The decomposition into a structured model provides clear divisions of the problem, well defined interfaces, a means to integrate and interrelate software components, a common language, and visibility of the design process. This is particularly valuable on medium to large

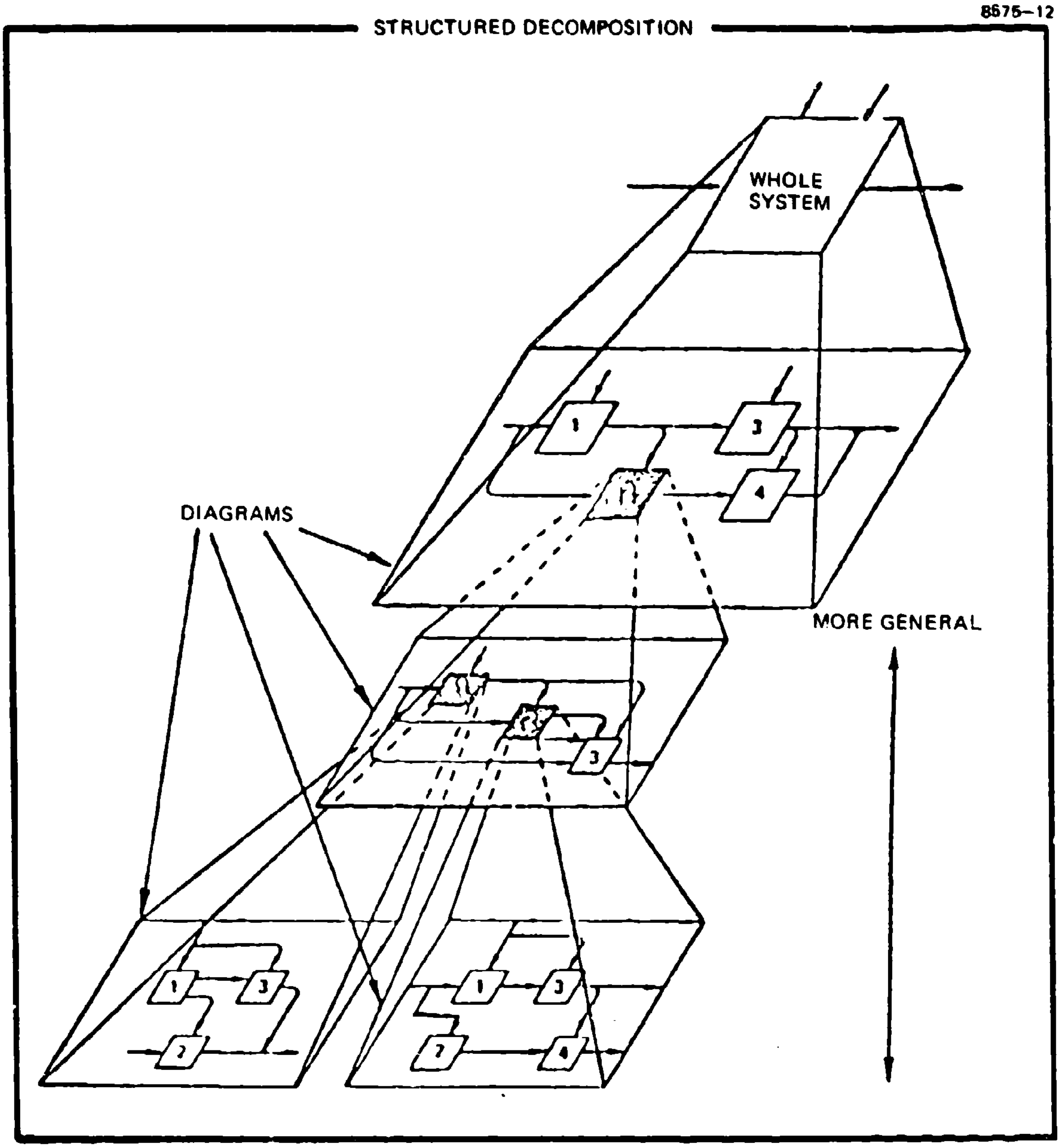


Figure 2-2. SADT decomposition.

projects in which several analysts or designers must work together effectively. This visible record of the design and its progress allow it to be iteratively reviewed. The review process is at least 30% of the power of SADT. In terms of project management, SADT models and their associated documentation and records provide a means to estimate code sizes, schedule required staff resources, assess design and implementation progress, ensure quality control through the review process, and ensure completeness and fidelity to the specified requirements. A summary of the specific HRL/SADT notation and diagram format is shown in Table 2-1 and Figure 2-3.

SADT is one of the best known software design methodologies. It was chosen here because we knew the most about it. An alternative called structured systems analysis (SSA) may be used at some point because of its more explicit representation. Once the design process has been completed, a language must be chosen to perform the implementation task. Naturally, this language must express the problem domain in a fashion such that its linguistic representation clearly specifies the logical intent of a program. This requirement provides capabilities whereby software systems can be better organized as they grow more complex, facilitate production of better programs with less effort, and ease program modification. Our language choice, PASCAL, meets this requirement, capitalizing on these features in addition to being compatible with standardization efforts such as ADA.

During the lifetime of a project, initial development costs account for only 25 to 33% of the total cost of a system; system maintenance and

Table 2-1. HRL/SADT Diagram Summary

- Component parts are shown as numbered boxes.
- A diagram may have no more than six boxes.
- Each box is detailed in one diagram at the next lower level until sufficient detail is reached.
- The place of each diagram in a model is indicated by a "node number": 21 is the diagram that details box 1 on diagram 2.
- The hierarchy of nodes is shown in a diagram index at the front of the package.
- Arrows represent constraint relationships among components (data or interfaces), not necessarily control flow or sequenc.
- The arrows show all that is needed by a box to perform its function.
- Arrows can be labeled I, C, O, or M to denote input, control, output, or mechanism.
- Numbers before ICOM labels refer to box numbers.
- Numbers after ICOM labels refer to relative position in parent box, with position one starting at the top left corner and increasing left to right, top to bottom.
- A, C, or I arrow on parent box is not limited to same role in detail diagram.
- An arrow unconnected in a detailed box, but not shown on the parent is surrounded by parentheses.
- A slash mark with a number on a path indicates the total number of lines that really follow a path.
- An* after a box number indicates that the box is decomposed to its lowest level in the diagram. The next form of description will be a PDL (program design language).

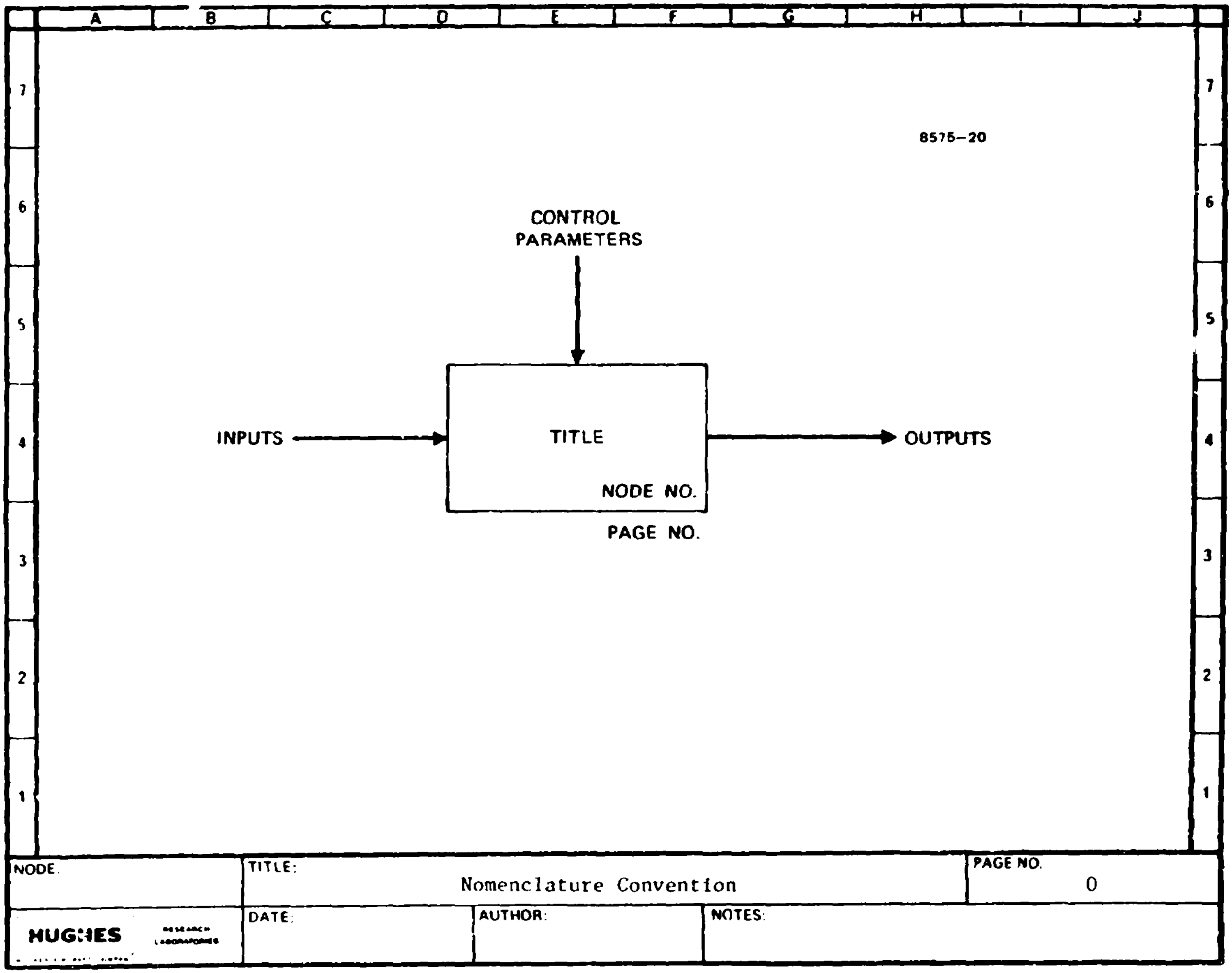


Figure 2-3. HRL/SATD diagram format.

operation are responsible for the remaining costs. This is especially true in a research environment, where a system is under constant scrutiny and subject to numerous changes, possibly because of dissatisfaction with system performance or failure to adequately achieve the system design goals. These changes can have disastrous consequences that were unforeseeable because of the complexity of a large system. PASCAL contributes to program readability and comprehension; it will not keep these "bugs" from occurring, but it will minimize the effort needed to detect and correct them. In addition, "bug" propagation will be minimized through program comprehension.

A current goal of high-level languages (HLLs) is to express algorithms clearly and precisely; HLL designers previously had not considered this to be necessary because the programs being constructed had been relatively simple. Designers had been more concerned with program efficiency because even the small problems being attempted taxed the limits of their computer technology. Now, however, since computer power has become abundant relative to human resources, HLL designers create languages that stress reliable and correct software at the expense of machine efficiency. PASCAL, a product of this trend, was designed as a machine-independent language that could express algorithms effectively. The current goal is to maximize a programmer's productivity by enhancing the programming medium. Thus, by using PASCAL, we hope to achieve the construction of large and complex systems in short time frames.

Structured programming is a disciplined approach to the implementation of complex algorithms, including limitations on allowable control structures and the use of abstraction, hierarchial structuring, and successive refinement to obtain better reliability, maintainability, and adaptability. There is nothing exotic about structure programming; it is simply the application of common sense organizing principles long used in hardware development and engineering in general. Although conceptually simple, such discipline has not been widely practiced in the construction of software. But when it had been used its success, as demonstrated by a growing number of users, suggests that we probably will gain by applying some of these techniques to our projects. To apply the implementation methodology effectively requires access to the structured control primitives. These are available in PASCAL but not in FORTRAN.

PASCAL encourages structured programming (not to be confused with goto-less programmin) by providing a small set of simple but robust controle and data structuring primitives nested inside one another. This improves readability and program comprehension by minimizing the number of interconnections between program parts and by so doing increasing reliability. PASCAL also encourages programmers to decompose systems into functional units. The advantage of function decomposition is clearly demonstrated by the numerous FORTRAN preprocessors now available that cascade the inner weakness of FORTRAN.

PASCAL encourages stepwise refinement (also known as top-down design), a process in which a goal or prohlem is stated as a subroutine of a single statement, which is then expanded into some number of basic control structures. At each level of description, the function is expanded into ever greater detail until the resultant program has been built. Thus, a program is built that is herarchically structured and described by successive refinements. Each refinement is described in terms of other refinements of which it is a component. This is not to imply that actual program conception proceeds in such a manner, only that it is desirable. Later refinements may be necessary when earlier decisions, when found inappropriate, are reconsidered. However, top-down programming does allow one to manage the undivided building blocks in an intelligent manner and to easily describe a system to another or to one's self.

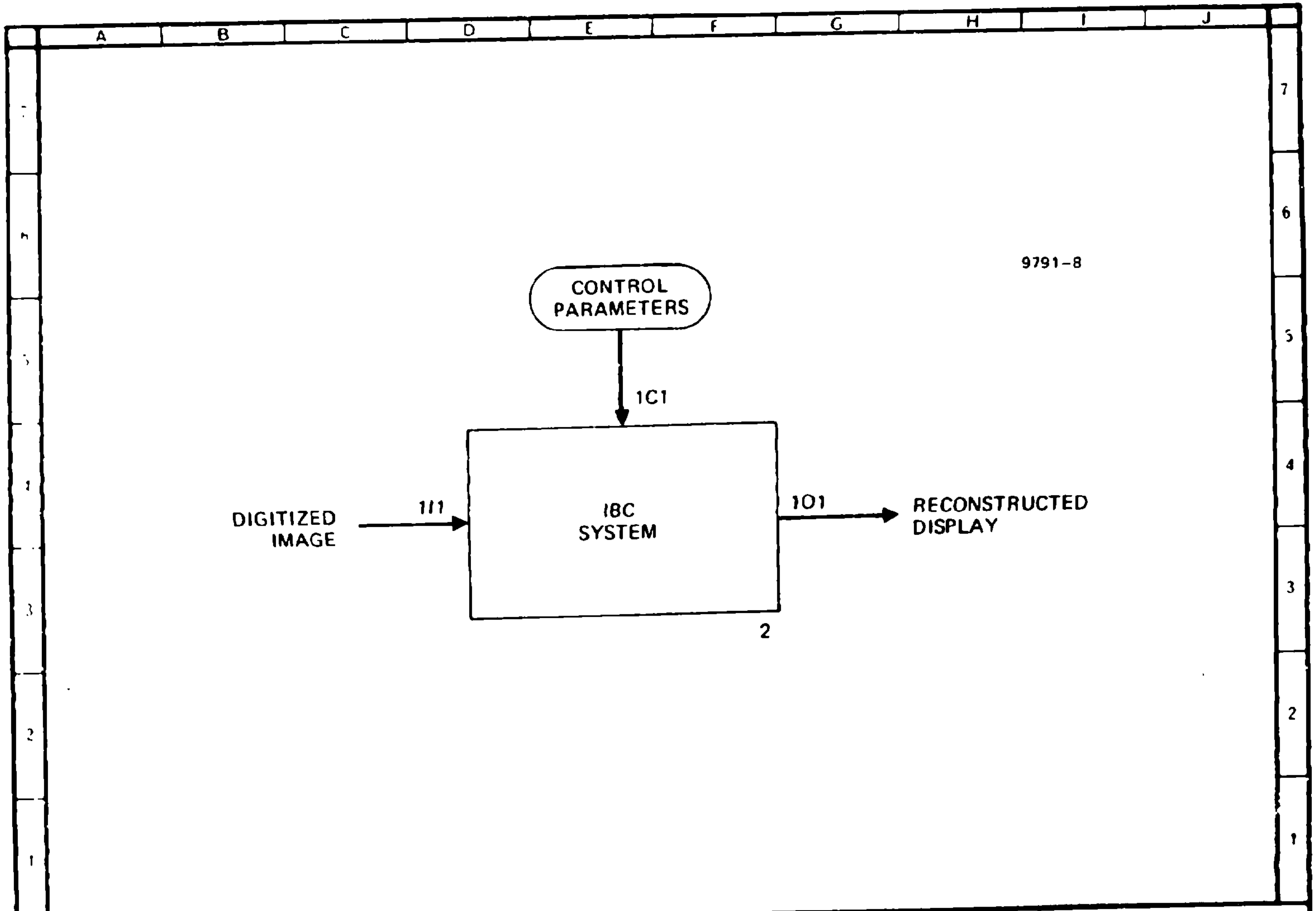
The efficiency of algorithms and data representations contributes more to program performance that do code optimization techniques. Therefore, we would like to use a programming language that can easily change algorithms and data representations; PASCAL is just that language. PASCAL provides strong structuring abilities, which allow a programmer to modify date representations without derastically changing the program code. This can be achieved using the techniques of levels of abstraction, information hiding, and module interfacing. Data are nothing more than a collection of objects, each with a set of allowable states. Procedures can be defined that cascade the representations of these objects in program modules. The user (of a program) then accesses objects by calling special procedures.

In addition, the richness of the PASCAL language eases the coding process and allows the programmer to devote more time to algorithm design during the coding process. The richness is provided by PASCAL's powerful control construct and data abstraction facilities.

The strategy presented here has contributed to achieving the goals of this project by utilizing the software engineering approach and capitalizing on the existing technology developed at HRL, which is embodied in a PASCAL base consisting of a body of PASCAL, FORTRAN, and assembly language programs.

C. TOP-DOWN IBC SYSTEM DESIGN

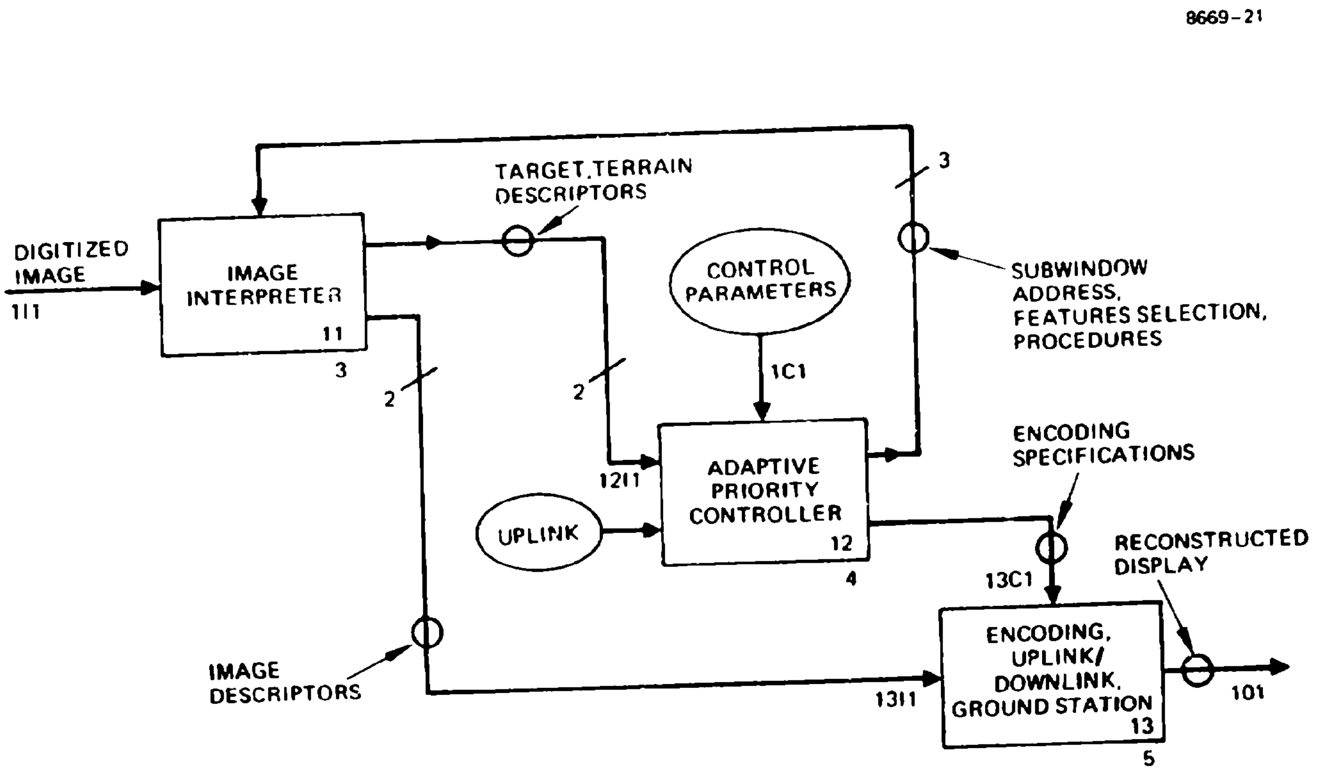
The top-down system design for IBC is shown in the following pages, using the SADT convention.



NODE 0	TITLE IBC SYSTEM		PAGE NO. 1
HUGHES RESEARCH LABORATORIES	DATE 4 JAN 79	AUTHOR TSENG/BULLOCK	NOTES

The scope of the IBC system is bounded by an input (III) which consists of a digitized image, an output (101) which consists of a bandwidth compressed data stream suitable for reconstruction of the input image, and a set of control parameters (ICI) which provide the strategy rules for target priority assignments and resource allocation of the image content consistent with the data link bandwidth available.

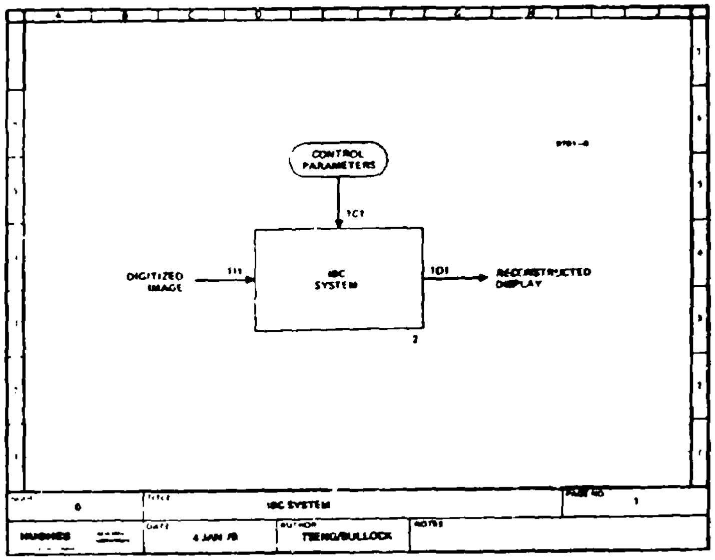
The input image is derived from the system sensor and A/D converted to the proper digital format for input (111) to the IBC system. The IBC system utilizes scene-analysis techniques to analyze the scene content with respect to targets and other key features. These targets and key features are then managed by a knowledge-based controller which assigns target priorities consistent with mission goals, provides dynamic resource allocation of available data link bandwidth to the feature and targets, and hands over the allocated resources for encoding and transmission to the ground station. The control parameters (ICI) provide a set of rules which is used by the knowledge-based controller. At the ground station, the output (101) of the IBC system is then used to reconstruct a bandwidth-compressed, synthesized image of the original sensed image.



NODE	TITLE	PAGE NO
1	IBC SYSTEM COMPONENTS	2

HUGHES RESEARCH LABORATORIES

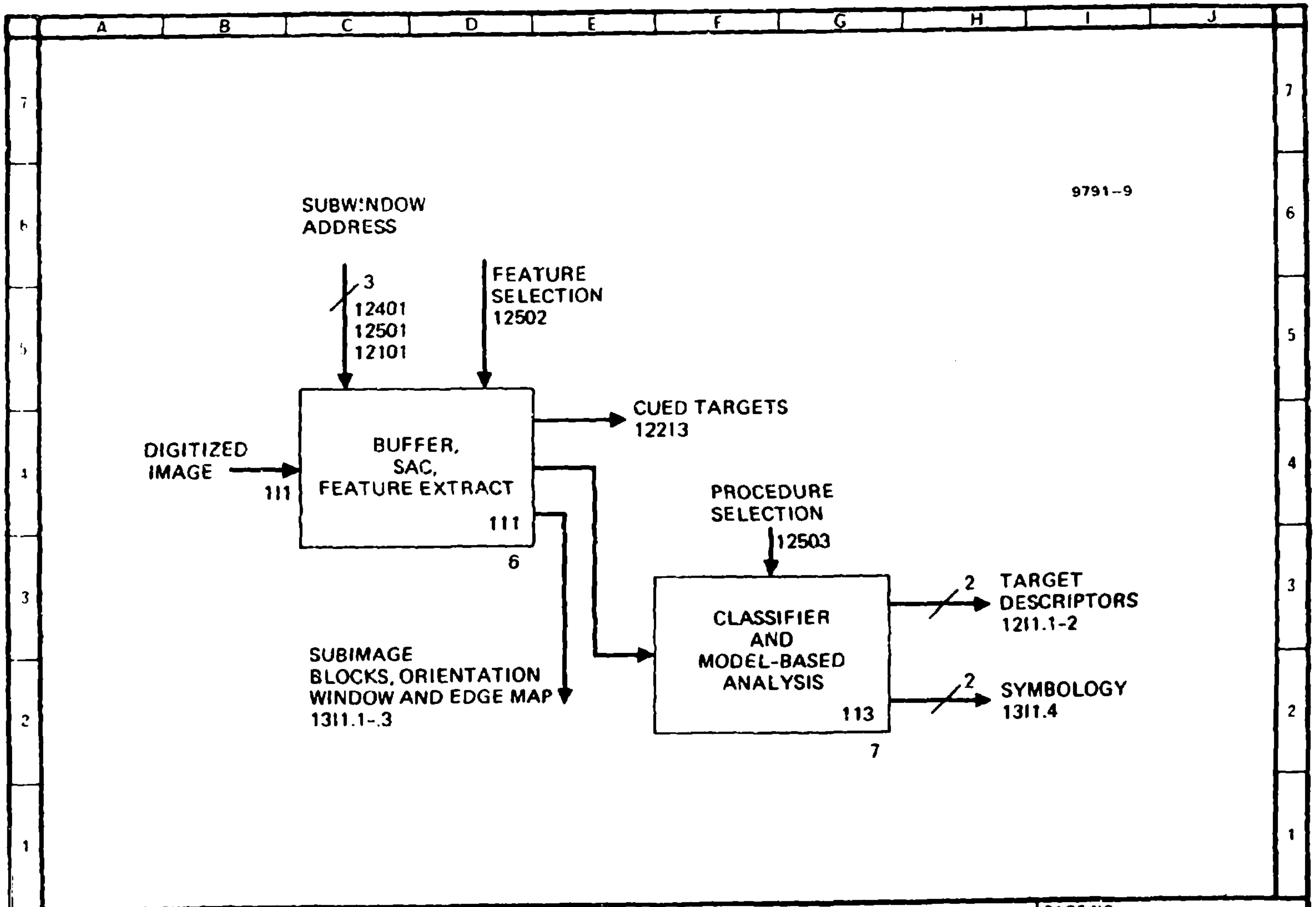
DATE	AUTHOR	NOTES
	TSENG/BULLOCK	



The IBC system decomposes into three components: image interpreter (11); adaptive priority controller (12); and encoder/decoder, up/down links, and ground station (13). The image interpreter (11) receives a digitized image (111) from the systems image sensor system. The interpreter performs three primary operations: target identification, statistical target (object) classification, and structural target (object analysis. Control information about where (addresses in the input image) and how (specification of features and analysis procedures to be used) the interpretation is to be performed is supplied by the adaptive prority controller. This particular control path is the most important in the entire system since it closes the control loop on the interpretation system, allowing iterative refinement of the results. The image interpreter provides two outputs: a list of target descriptors that goes to the adaptive priority controller and a collection of image descriptors that goes to the encoder/decoder.

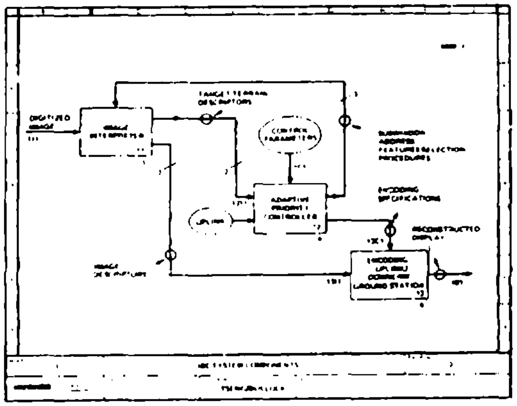
Three types of target descriptors can be found in the list going to the priority controller: an indication that a target has been identified (cued) at a given location, an initial classification, and a refined structural scene analysis classification. The adaptive priority controller (12) uses these inputs to determine what typeof processing should be used and what the image interpreter (11) should do next. The control policy for this decision is contained in a table of control parameters that lists the target and scene object processing priorities and the allocations allowed for different scene components for the encoded signal to be transmitted. This table of control parameters can be changed dynamically by the operator to suit his needs through the uplink input.

The image descriptors from the image interpreter consists of edge image subwindows and symbolic target/object descriptors. These are encoded by the encoder/decoder (13) for the transmission through a simulated downlink (with jamming and noise), decoded and reconstructed in the simulated ground station, and displayed. This displayed image is the 1000:1 bandwidth compressed version of the sensor scene.



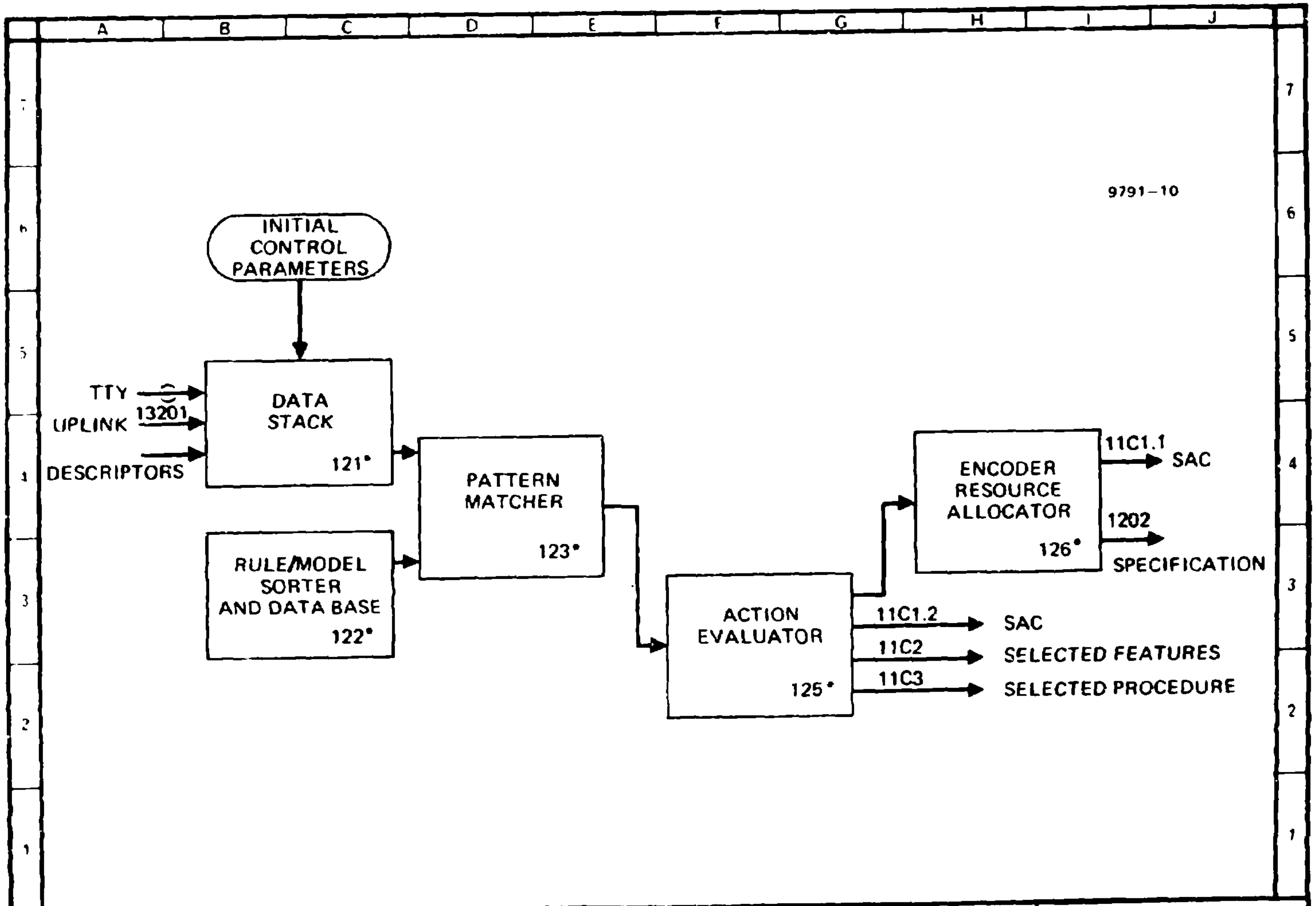
NODE 11	TITLE IMAGE INTERPRETER		PAGE NO. 3
HUGHES RESEARCH LABORATORIES	DATE: 30 MAR '79	AUTHOR: BULLOCK	NOTES: REV-3

The image interpreter (11) accepts the digitized input image and produces a symbolic description suitable for the mission goals. There are two components: (1) the buffer, subwindow address controller (SAC), and feature extractor (111) and (2) the classifier and model-based analyzer (113).



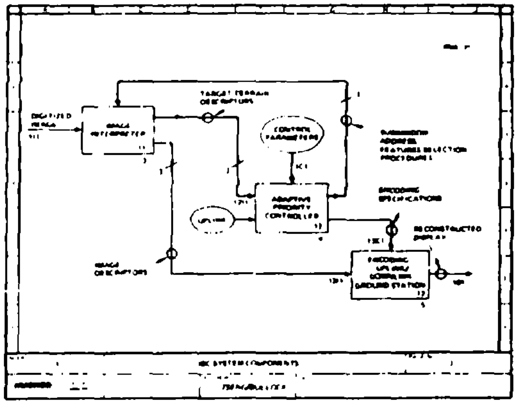
The buffer in (111) is a full-frame software buffer that stores the input image during the entire processing cycle. This allows the interpreter to produce detailed analysis results through a process of iterative refinement. The subwindow address controller (SAC) receives requests from the adaptive priority controller (12101) in the form of addresses to interpret the image in a subwindow at different specific locations. In the initial pass, these locations are sequentially contiguous and cover the whole image. In later refinement passes, the processing will uusally be at selected locations. The feature extraction includes processes for deriving low-level statistical features (mean, standard deviation, etc.) and high-level features (lines and boundaries, vertices, shape moments, etc). The order and types of the features produced is hardwired on the first pass but can be modified during refinement passes by inputs (111I2) from the adaptive priority controller. One output is the set of derived features (11101) that goes to the classifier and model based analyzer (113I2). The other is a collection of sampled windows centered around targets and a low resolution (sampled) subwindow centered over the highest priority target for orientation purposes (11102) for downlink transmission.

The classifier and model-based analysis system accepts the scene features (113I2) and derives symbolic target and terrain labels based on statistical and structural properties of the features. The correspondence between these properties and the labels is stored in a set of structural scene models and a statistical training set 113I1. Specific matching techniques can be selected by control (113C1) from the adaptive priority controller. The outputs consist of a list of labeled target descriptors (11301), which is used by the adaptive priority controller to determine the analysis refinement process, and a collection of labeled object symbology for downlink transmission (11302).

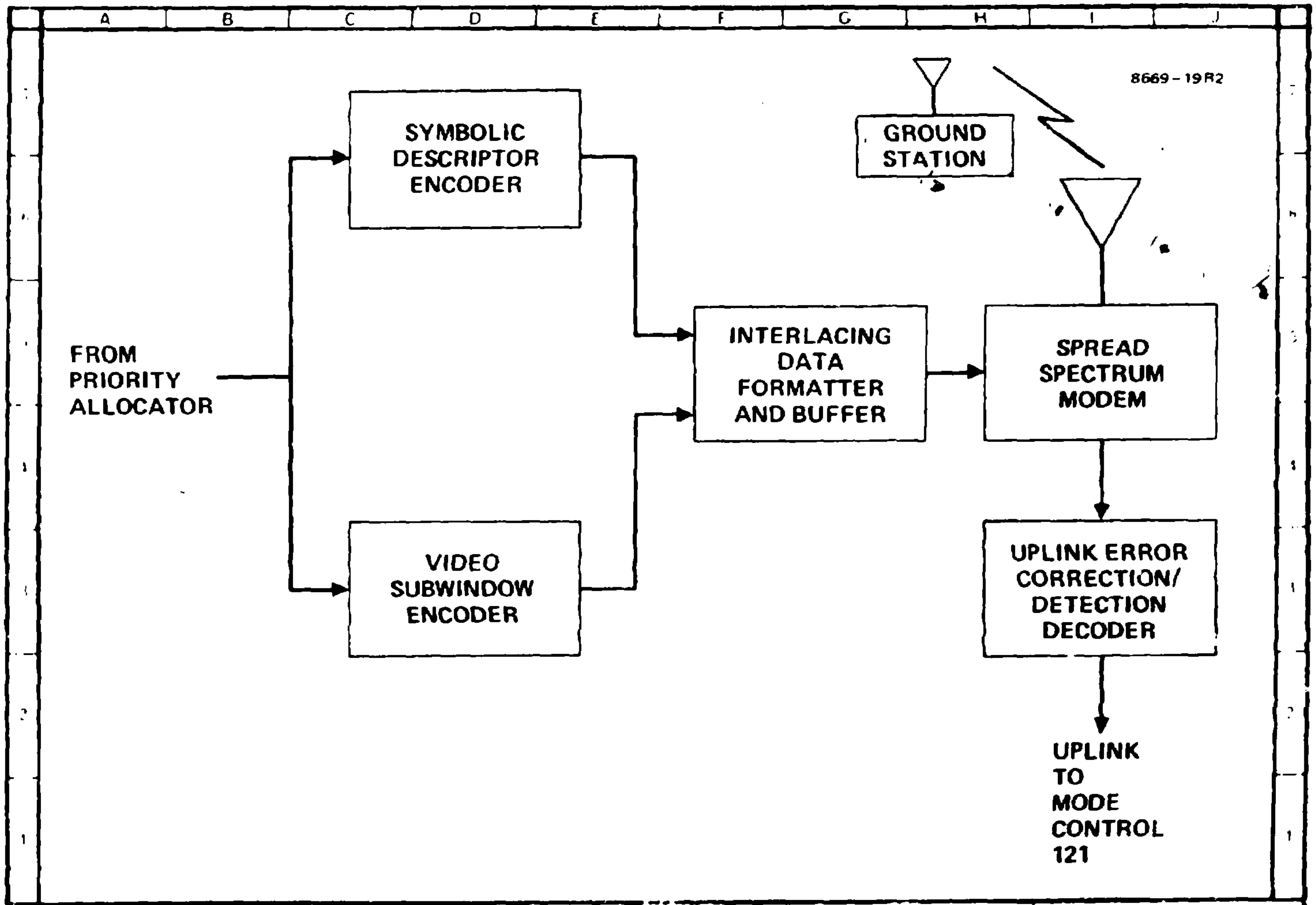


NODE 12	TITLE (ADAPTIVE PRIORITY CONTROLLER)		PAGE NO. 4
HUGHES RESEARCH LABORATORIES	DATE 26 MAR 79	AUTHOR BULLOCK	NOTES: REV-2

The adaptive priority controller (APC) functions as the overall system controller. Based on the user's current goal and the desired analysis results at a given time, the APC determines the next analysis step. The APC is organized as a knowledge-based production system in that its operation is determined by several rules and symbolic models stored in a data base (122). The rule format consists of two parts: a condition, an action, and a priority. Through a process of pattern matching (123), the compound condition predicates (a conjunction and disjunction of atomic conditions) in the rules are matched against an input data item until a match is found. The input data items include symbolic image analysis results (edges, lines, regions, objects, etc.), *uplink commands* (13201) *to modify* the operation of the APC by editing rules or changing parameters, and TTY commands used to simulate the uplink during development. The data items are stored in a FIFO queue (121) based on their order of entry. The initial default condition of the APC parameters are input to this queue at startup to initiate the operation.


DIGITIZED IMAGE
IMAGE INTERPRETER
TARGET TERRAIN DESCRIPTORS
CONTROL PARAMETERS
SUBWINDOW ADDRESS FEATURES SELECTION PROCEDURES
ADAPTIVE PRIORITY CONTROLLER
UPLINK
ENCODING SPECIFICATIONS
RECONSTRUCTED DISPLAY
IMAGE DESCRIPTORS
ENCODING UPLINK DOWNLINK GROUND STATION
SYSTEM COMPONENTS

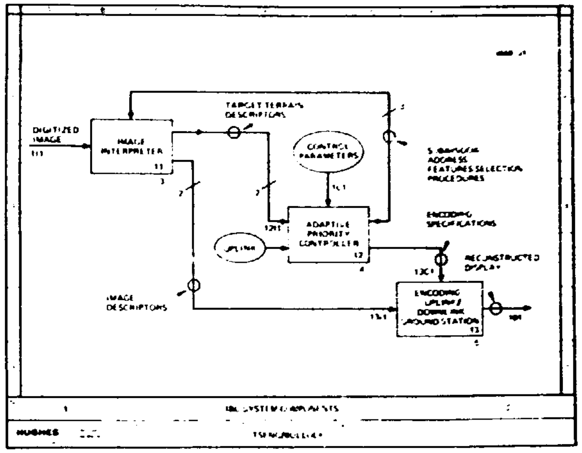
When a rule condition matches in input data itme, the action portion of the rule is sent to the ACTION evaluator (125). The evaluator interprets the (compound) action and sends commands to the appropriate analysis system module to carry out the desired operation. These commands can write high level derived object descriptions into a descriptor data base (126); position a subwindow for processing 11C1, 2 (SAC); select the best subset of features to be extracted (11C2); and select the decision process (11C3), enter new production rules or edit their contents. As objects are identified, they are enter[illegible] into the Encoder Resource Allocator (126). This allocator contains a table which tells how many of each type of output result (target images, edge image, symbols, etc.) are to be encoded for downlink transmission (1202). The allocator constrains the quantities so that the channel capacity is not exceeded. The allocation table can be changed by the actionof a rule directed from the uplink (or TTY).


8669-19R2
FROM PRIORITY ALLOCATOR
SYMBOLIC DESCRIPTOR ENCODER
VIDEO SUBWINDOW ENCODER
INTERLACING DATA FORMATTER AND BUFFER
GROUND STATION
SPREAD SPECTRUM MODEM
UPLINK ERROR CORRECTION/ DETECTION DECODER
UPLINK TO MODE CONTROL 121

NODE 13	TITLE RPV ENCODERS/DECODERS AND MODEMS		PAGE NO 5
HUGHES RESEARCH LABORATORIES	DATE 12 APR 1979	AUTHOR BULLOCK/TSENG	NOTES REV 1

PRECEDING PAGE BLANK-NOT FI

This system accepts a mixed collectio of allocated image descriptors. This information is routed to an appropriate encoder, which performs optimal encoding for a given type of data (symbols, edges lines, and gray-level subwindows). The encoded information is formatted and buffered for transmission by a modem. The modem also receives downlink commands and sends them to the mode control.



SECTION 3

IMAGE INTERPRETER

A. AUTOMATIC TARGET CUEING

The auto-cuer incorporated in the image interpreter component of the IBC system consists of the software simulation algorithm developed in the ATAC program. This auto-cuer had originally been designed for use with FLIR (infrared) imagery. The minor modifications needed (because of the visible band IBC data base imagery used) to adapt it involved mainly the low-level processor, where new weighting parameters for the linear discriminant of the auto-cuer had to be selected. The weighting factors were redistributed to reflect the better edge-definition targets contained in the visible band data base imagery as compared with the ATAC FLIR imagery. The salient features of the ATAC auto-cuer are presented here. Automatic target cueing of FLIR imagery has received much attention recently, and a representative literature list is given in Refs. 1 through 12.

The auto-cuer simulation utilizes a two-level approach to automatic target cueing. A low-level processor examines the full-frame image and selected points of interest where targets are likely to be located. These points of interest are then sent to the high-level processor for further, and more detailed, examination in window regions centered about the interest points. Here, object segmentation and feature extraction take place, and targets are detected and identified by the classifier.

In determining the low-level points of interest, a small number of easily calculated statistical parameters of the image are considered. The motivation here is to form, by a proper choice of low-level parameters, a linear discriminant of these parameters so that target areas are more likely to possess high discriminant values. By appropriately thresholding the discriminant at this stage, a small number of interest points are produced and passed on to the high-level stage for further processing. The remaining, below threshold, points sufficiently far from interest points are excluded from further consideration and processing. Theobjective of the low-level processor is to locate as many of the target/areas as possible using interest points, while discarding as many areas of the image as possible from complex and time-consuming calculations by the high-level processor.

At the high-level stage, a window is centered about each interest point in the image, and objects from the background, or clutter, are extracted with an object semgentation algorithm. Subsequently, all segmented objects are characterized by several high-level features. Finally, all the segmented objects are sent to the high-level classifier, where targets are detected and identified and clutter is rejected.

1. Low-Level Interest Operator

The function of the low-level stage of the auto-cuer is to select and isolate those points of interest in the image where targets are most likely to be located. Low-level processing, as defined here, is the process by which each 5 x 5 pixel region of the shrunk image is assigned a value, via the linear discriminant, that is intended to be rperesentative of the probability that this region is partially or totally contained within a target. The objective is to utilize only a small number of easily calculated, statistical parameters of the image to extract a high percentage of the targets, by means of the derived interest points, while excluding a large portion of the image from further consideration by the high-level stage. The selectionof the features used in the low-level processing were chosen not only for their usefulness in differentiating targets from the background, but also for their potential ease of implementation. The low-level features used in this auto-cuer simulation study are shown in Figure 3-1. As the figure shows, a local window consists of a 5 x 5 pixel region, a large local window consists of a 60 x 60 pixel region, and global values are taken over the entire image. These ten features are combined by the scalar to form eight contrast- and offset-invariant features. These features are then used to form a weighted sum for each small window (5 x 5 pixel) region. The small windows of the low-level processor can be made to slide in an overlapping fashion or can be taken in contiguous sectors. The degree of overlap is variable and can range from 1 to 5 pixels, resulting in a low-level feature picture of the same size or one shrunk to a maximum of 1/5 the size of the original image. The degree of overlap can also be taken to be independent in the two directions. The weighting coefficicents used in our simuiations are shown in Figure 3-1.

The value of the linear discriminant, formed from the weighted linear combination of features F_1 through F_8, gives a measure of the

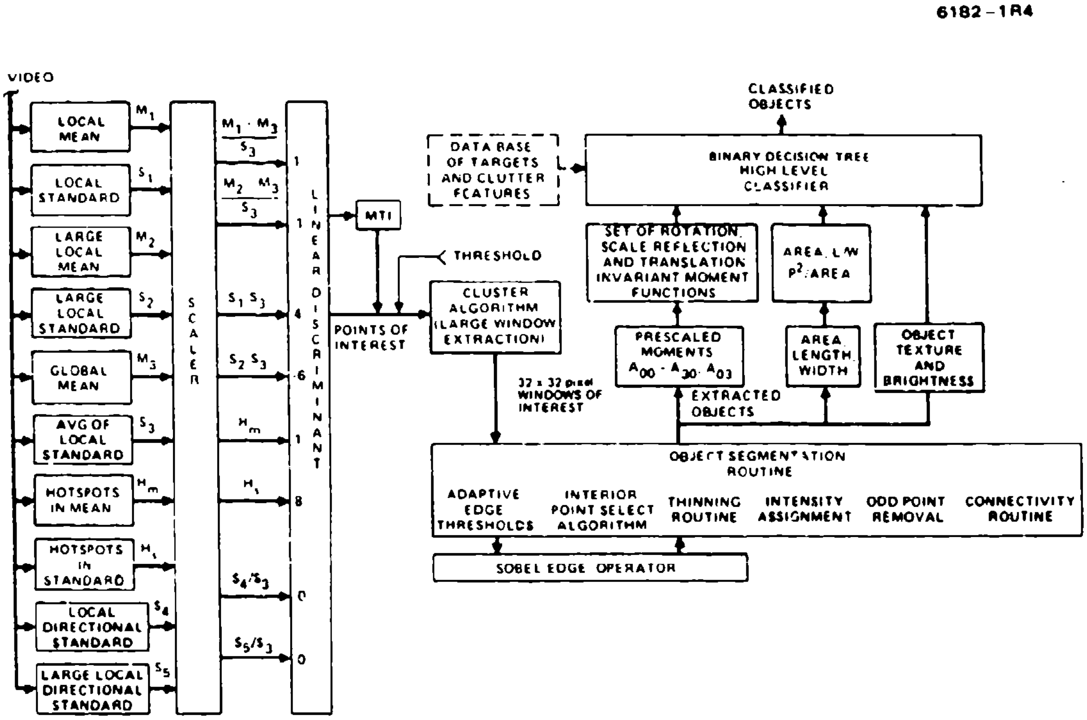


Figure 3-1. ATAC auto-cuer block diagram.

probability that this small window region is partially or totally contained within a target. That is, high-linear-discriminant-valued small windows are more likely to be contained within targets. At this stage, a threshold is set, and small windows that have linear discriminant values above threshold are chosen as regions likely to be within targets. Those windows are labeled as interest points and tagged for further processing by the high-level stage. All other points below threshold and sufficiently far from interest points are excluded from further consideration and processing. The relative values of the weighting coefficients used for features F_1 through F_8 are shown in Figure 3-1.

In addition to the low-level features mentioned above, a moving target indicator (MTI) which detects changes in either the original image or the low-level interest points from frame to frame can be incorporated into the linear discriminant, as shown in Figure 3-1. This feature has not yet been implemented in the simulation studies.

2. Object Segmentation

After interest points are identified by the low-level processor, regions of interest (defined by large windows 32 x 32 pixels in area) are centered about the interest points. The extraction of targets from the background, or clutter, takes place in the segmentation routine. It is the function of the segmentation algorithm to separate those areas believed to be interior to an object of interest from the background enclosed by a large window (region of interest). The accurate segmentation of objects from the background constitutes a critical stage of the auto-cuer simulation. Reliable high-level features that characterize target and clutter objects can be derived only if accurately segmented targets are extracted.

The initial step in the segmentation routine is the generation of an edge value map within the region of interest. For this purpose, a modified Sobel operator of 3 x 3 pixel size was used in the simulation studies. The form of the Sobel operator is shown in Figure 3-2. The (Sobel) edge values corresponding to targets are assumed to be among the largest edge values in the region of interest. As used in this algorithm, an object of interest is a contiguous area consisting of interior points. An interior point is defined as a pixel that is surrounded in most directions by edge points, an edge point being defined as a point with an associated edge value that is above some threshold value.

Edge threshold value is determined adapitvely using the histogram of edge values within the region of interest. Based on the expected range of target sizes, the 80th (E_{80}) and 95th (E_{95}) percentile edge values within the windows are used as control end point values. In addition, two absolute (arbitrary) edge values $T_1 = 100$ and $T_2 = 50$ are also chosen. To determine the threshold value for a given region of interest, E_{80} is compared to T_1. If E_{80} is greater than or equal to T_1, then the threshold is set to E_{80}. If E_{80} is less than T_1, then the next higher intensity value within the region of interest is picked (E_{next}), and its corresponding percentile (P_{next}) is noted. An expression

$$T_\Delta = \frac{T_1 - T_2}{P_{95} - P_{80}} (P_{next} - P_{80})$$

76101

A	B	C
D	E	F
G	H	J

$$\nabla x = (C + 2F + J) - (A + 2D + G)$$

$$\nabla y = (G + 2H + J) - (A + 2B + C)$$

- SOBEL EDGE GRADIENTS, ∇x AND ∇y, ARE EXTRACTED WITH 9-ELEMENT WINDOW CENTERED AT PIXEL POSITION E.

- EDGES ARE ESTABLISHED BY THRESHOLD TEST OF EDGE MAGNITUDE, $|\nabla x| + |\nabla y|$

  IF $\frac{|\nabla x| + |\nabla y|}{T} > 1$, EDGE IS ESTABLISHED AT PIXEL POSITION E.

- EDGE ORIENTATION, $\theta = \tan^{-1} [\nabla y/\nabla x]$.

Figure 3-2. Sobel edge extraction.

is computed and compared to E_{next}. If the current E_{next} is greater than or equal to T_{Δ}, then the threshold is set to E_{next}; if not, the process is repeated by picking the next higher intensity level. Finally, if E_{95} is reached, it is compared to T_2. If E_{95} is less than or equal to T_2, then the threshold is set to T_2. Otherwise, the threshold is set to E_{95}. In this way, the edge threshold is set adaptively in each region of interest, depending on the target size, presence of other strong edges, and the amount of noise present.

Once the above-threshold edge points have been determined, the detection of interior points within the region of interest takes place. An interior point is (arbitrarily) defined as a pixel that is surrounded in six of eight directions by above-threshold edge points. A thinning or filling operation then removes isolated interior points and fills in gaps in the binary interior point image. The thinning and filling operator consists of a 3 x 3 pixel overlapping window, which is slid within the region of interest in both the x and y directions. If five out of nine pixels in the 3 x 3 window are occupied by interior points, then the center element of the window is set to 1; otherwise, it is set to 0.

Because of the coarseness of the eight-direction edge point search, concave and convex portions of objects tend to be filled in, or shadowed; the same is true for regions between two objects that are close to each other. To overcome this drawback, a maximum-likelihood assignment of interior versus exterior points is carried out next. To do this, histograms of the previously derived interior and exterior regions are tabulated. Then, the intensity of each pixel in the interest window is compared to the two histograms. The pixel in question is now assigned as an interior or exterior point based on the maximum likelihood that this pixel intensity belonged to one of the two histograms. After this intensity assignment, another thinning and filling operation is performed, again to eliminate isolated interior points or fill in gaps in the newly formed objects. The thinning and filling operator is identical to the one described above. Subsequent to the second thinning and filling operation, a connectivity routine is used to merge those interior regions that are likely to be connected, but were segmented as separate areas because of, for example, gaps or degradations in the object caused by noise or reflective differences in the target.

Next, the connected interior point regions thus derived are compared with the first interior point classification in the interest window. Those areas having the most coincidence of interior points with the original interior point map are designated as segmented objects, and their corresponding original intensity values are substituted into the segmented objects. Depending on the scenario expected in the images, one or more of the segmented objects within the interest window can be tagged as likely targets. In this simulation study, the area with the largest coincidence was chosen as the segmented object. Finally, a calculation is made (based on area and object center) to see if it is likely that portions of the desired target lie outside the interest window. If so, the window is repositioned and the segmentation is recomputed. These segmented objects are passed on to the high-level feature extractor for classification.

3. Extraction of High-Level Features

Once object segmentation has been completed and interior points have been separated from background clutter, a high-level feature vector is calculated to characterize each segmented object within an interest window. These high-level-feature vectors are used subsequently in the binary decision tree classifier for target detection and recogniation and clutter rejection. One purpose of the features is to reduce the dimensionality of the decision regions from the total number of degress of freedom of the pattern to the number of features extracted. A second purpose is to improve the performance of the classifier by only allowing relevant pattern parameters to influence its training.

The high-level-feature-extraction algorithm first detects all interior points as determined by the segmentation routine and replaces each detected interior point by its original pixel intensity. All other points within the interest window are set to zero. The centroid of the segmented object is then determined, and a set of moments (up to and including third-order moments) is calculated relative to the centroid, with distances scaled to the square root of the extracted area. These moments are then transformed to a set of seven moment transforms, M1-M7, which has the property that it is invariant to rotation, reflection, translation, and scale. Additionally, the calculated moments are used to derive the length, width, and aspect (length/width) of the segmented object. Finally, a measure of the object intensity against background is also generated. The collectionof all the feature vectors for the segmented objects of the data base are then used for the training, detection, and classification of targets.

4. Target Detection and Recognition

After the segmented objects have all been characterized by feature vectors, they are ready for use in the high-level classifier. The function of the high-level classifier is to perform clutter rejection on the segmented objects and recognition on the detected targets. A binary decision tree classifier has been used to detect and recognize targets based on their feature vector characteristics. For targets segmented into single objects, the conventional binary decision tree classifier has performed well. However, when single targets are segemented into multiple objects, the conventional classifier fails because it does not associate the multiple objects as a single target.

We have extended the ATAC auto-cuer to include a model-based classifier, based on previous work developed on IR&D projects and other contracts (e.g. DARPA Autonomous Terminal Homing program). The auto-cuer was originally designed and developed for thermal IR imagery. In applying the auto-cuer to visible imagery, we found that statistical target classification was not always an adequate measure. Model-based classification techniques have proven useful in eliminating clutter and increasing classification certainty.

Currently, our model-based classifier can identify trucks and jeeps. It does so by locating the regions corresponding to the hood and the cab roof. In the visible imagery, both of these areas represent high reflectivity regardless of sun angle. In the conventional auto-cuer, the target would be segmented as two separate regions. The shape of the individual region is regular, but the size is small enought that the object would probably be discarded as clutter. However, with the knowledge-gased system, our model-based classifier characterizes truck and jeep targets as two regularly shaped regions within a defined area and distance tolerances.

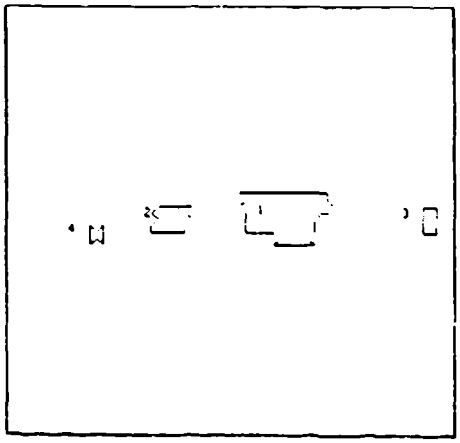
The first step in building the model is to invoke a smoothing operator over the high-level window encompassing the potential target. This is done to remove as much noise and jitter as possible so as to obtain a regular object. Next, a region operator is applied to the window to find areas of uniform intensity within a minimum area requirement. The mininum area value is a function of the approximate range to the object. The regularity of each region is measured by comparing the true region area to that of the best fit rectangle. If the area of the region is less than 50% that of the rectangle, the region is discarded.

The regions that pass the minimum-area and regularity tests are eligible for model-based classification. Two other properties must be satisfied for the two regions to be classified as a truck or a jeep:

- Areas of the smaller region to be within 80% of the larger
- Distance between the centroids must be less than d, where d is dependent on the approximate range to the target.

If two regions satisfy these and other conditions then the window is labeled as a truck. Whenever more than two regions within the window meet the classification requirements, the window is identified as clutter. Examples of region extraction and classification are included in Figures 3-3 to 3-5.

9008-14



64 x 64 WINDOW

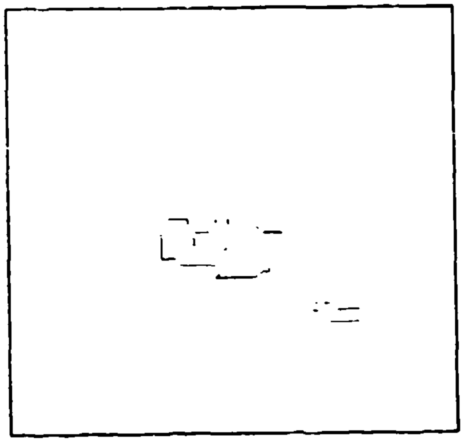
REGIONS 1 AND 2 CLASSIFIED AS TRUCK

REGION	PERIMETER	AREA	CENTROID (x, y)
1	44.1	80.0	40.7, 31.5
2	20.5	21.0	23.9, 32.0
3	12.0	9.0	62.0, 32.0
4	11.7	5.0	13.0, 34.5

REGIONS 1 AND 2 CLASSIFIED AS A TRUCK

Figure 3-3. Example of model-based classification to identify a truck.

9008-17



64 x 64 WINDOW

REGION	PERIMETER	AREA	CENTROID (x, y)
1	57.5	112.0	30.6, 36.2
2	16.5	12.0	48.6, 46.7

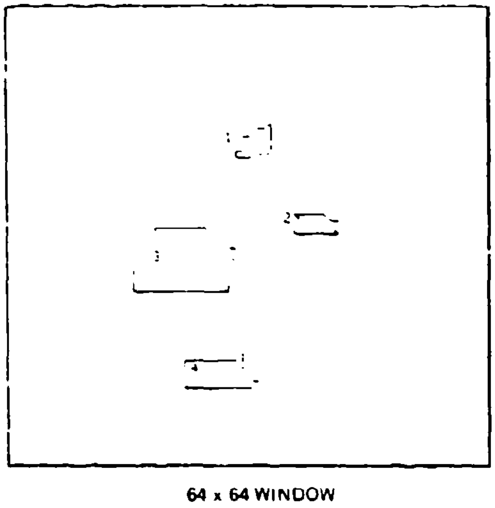
REJECTED CLASSIFICATION BECAUSE

(a) AREA RATIO BETWEEN REGIONS 1 AND 2 GREATER THAN TOLERANCE

(b) DISTANCE BETWEEN CENTROIDS GREATER THAN TOLERANCES

Figure 3-4. Example of model-based classification to reject clutter.

9008-15



REGION	PERIMETER	AREA	CENTROID (x, y)
1	19.9	15.0	33.8, 19.0
2	19.1	16.0	41.8, 30.6
3	44.4	108.0	23.8, 36.2
4	31.9	35.0	29.0, 52.1

REJECTED CLASSIFICATION BECAUSE (a) MULTIPLE TARGETS WITHIN WINDOW (REGIONS 1 AND 2 CLASSIFIED AS A TRUCK ALSO REGIONS 3 AND 4 CLASSIFIED AS A TRUCK

Figure 3-5. Example of model-based classification to reject clutter.

Model-based classification is a natural extension of the conventional auto-cuer in a knowledge-based system. We have applied the algorithm to several images in the data base and have correctly classified all seven truck and jeep targets with only one false alarm.

B. EDGE-MAP EXTRACTION

Edge maps were derived from the Nevatia-Babu line-detection algorithm. The edge detection is done by convolving a given image with 5 x 5 masks corresponding to ideal step ecges in six directions (0°, 30°, 60°, 90°, 120°, 150°). The magnitude of the convolved output and the direction of the mask giving the highest output at each pixel are recorded as edge data.

The edges are next processed through a thinning and thresholding procedure. The presence of an edge at a pixel is decided by comparing the edge data with some neighboring pixels. An edge element is said to be present at a pixel if:

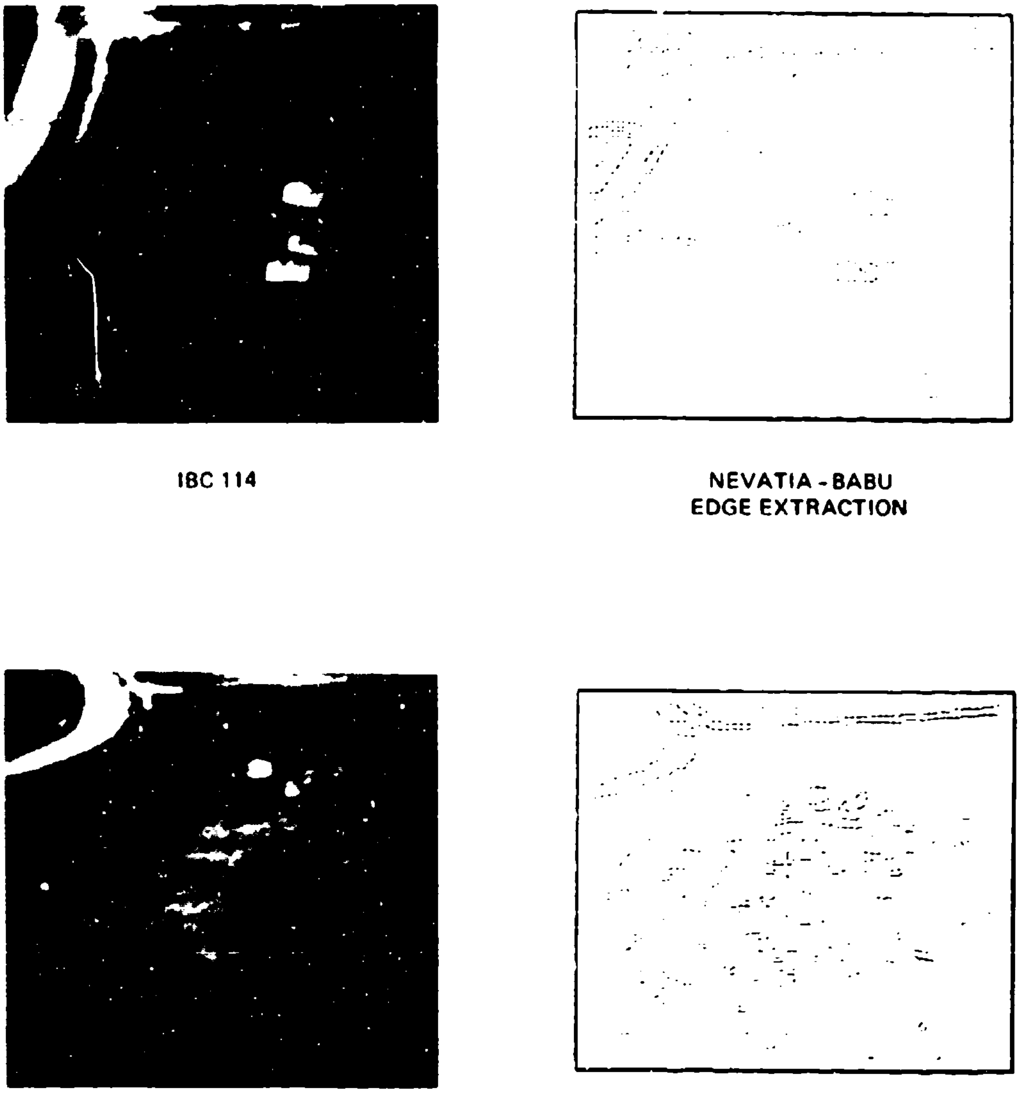
(1) The output edge magnitude at the pixel is larger than the edge magnitude of its two neighbors in a direction normal to the direction of this edge.

(2) The edge directions of the neighboring pixels are within 30° of that of the central pixel

(3) The edge magnitude of the central pixel exceeds a fixed threshold.

If conditions 1 and 2 are satisfied, the two neighboring pixels are disqualified as edge candidates. The threshold value in condition 3 is an absolute value and is therefore set to a minimum in an attempt to make the edge operator tolerance to intensity differences between images in the data base. To slect the dominant edges, a second thresholding is performed on the entire edge picture after the completion of the Nevatia-Babu edge extraction algorithm. A histogram is calculated for the edge magnitudes, and the strongest 5% of the edges are retained. Examples of the resulting edge maps are shown in Figure 3-6.

9008-1R1



IBC 114

NEVATIA - BABU
EDGE EXTRACTION

IBC 209

NEVATIA - BABU
EDGE EXTRACTION

Figure 3-6. Edge map extraction base on IBC data.

SECTION 4

KNOWLEDGE-BASED CONTROLLER

For the most part, scene-analysis systems have had hard-wired hierarchical organizations. Such programs suffer from fragility in the operating domains due to noise, texture, shadows and shape variations. One apporach towards dealing with these problems is to collect a great deal more information about the scene from multiple feature sources (edges, regions, texture, etc.) and to interpret the information in terms of more sophisticated and integrated models.[13-15] The inexact, and error-filled information of previous systems is thus supplemented with redundant information, and the fragile processing algorithms are replaced with systems of algorithms that can piece together the fragments into the desired results.

Research into the construction of scene-analysis systems with these improved capabilities has gone on at HRL since 1974. This work has concentrated on the use of production systems and, more recently, more sophisticated knowledge-based systems as an organizational and control framework. A production system is a powerful, yet deceptively simple, symbolic process manager. Production systems have recently been undergoing a period of rapid development for complex problem sovling applications.[16]

In its simplest form, a production system consists of a set of rules, a data base, and a rule interpreter. The rules are in he form of rewriting rules that have left side conditions and right side actions:

PRECEDING PAGE BLANK-NOT FILMED

(RULE NAME: ((condition))((action))

or simply: (condition) (action)

The data base is, in fact, a world model that contains the systems' current view of its situation or interpretation of the visual features. The basic function of the production system interpreter is to match the current world model situation against the conditions in the set of production rules. When a match is found, the interpreter evaluates the corresponding action specification, resulting in a modification of the data. This usually implies placing a new symbol in the world model data base, erasing an old symbol, or executing a process. Example shown in Figure 4-1.

The production system methodology is ideal for organizing a flexible system to deal with a large amount of inexact, error-filled but redundant scene information. Traditional fragile version systems are a collection of procedures and models carefully linked by the programmer with hardwired procedure calls. Using the production system it is possible to build a system consisting of a (possibly large) number of highly independent operator and interpretation modules thatonly communicate when necessary. The activity of the modules is actually data- or event-driven based on the effect of changing system goals and the chaging input scene. Each module can be executed at any given time based on its statement of relevance (the condition part of its associated rule).

This sectio describes the production system, rules, and system operation. The production system and rule set have been kept very simple in order to demonstrate feasibility without getting into an implementation trap. Thus, a great deal of elaboration and improvement is possible in the future. The system wasl also kept simple because it was our first

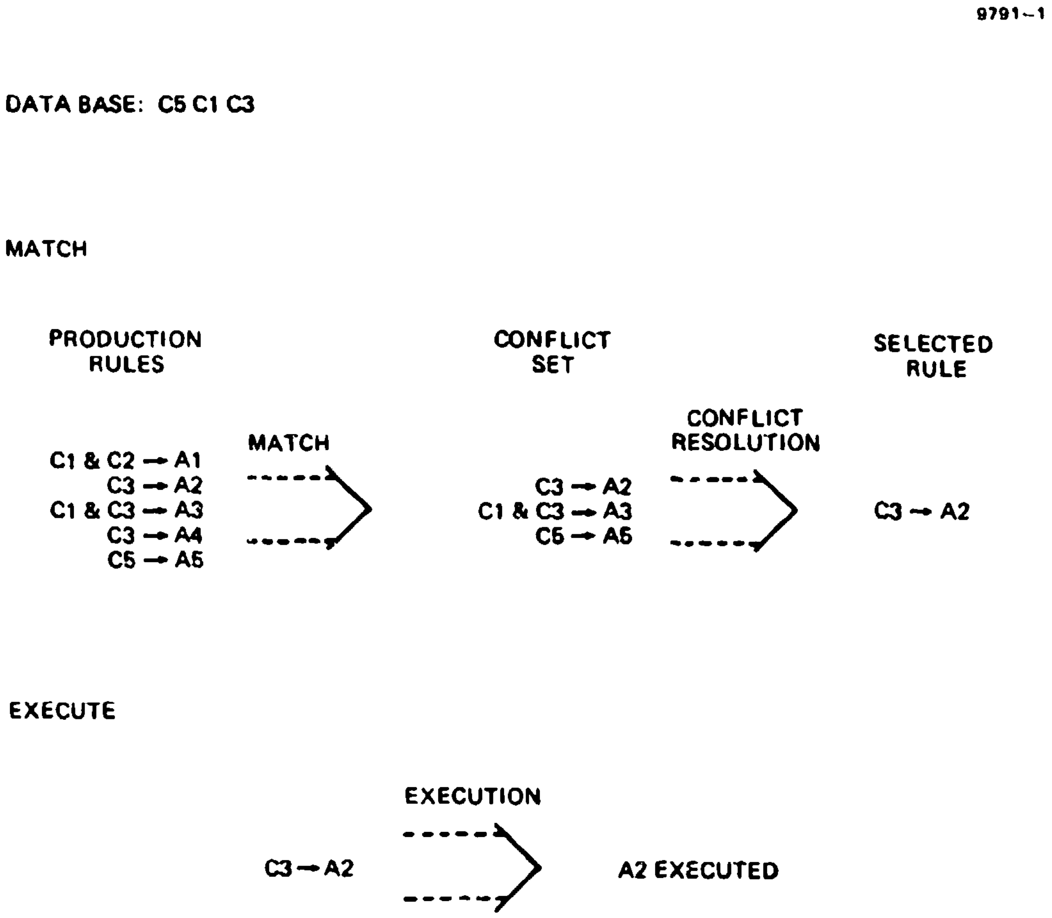


Figure 4-1. Basic production system operation.

attempt at a (non-LISP) PASCAL based production system. We believe that a non-LISP system of this type may be more practical for future limited-resource embedded computer applications.

A. PRODUCTION SYSTEM

The production system consists of a:

- Blackboard
- Set of production rules
- Interpreter.

The blackboard contains all the known facts during execution of the production rule system (PRS). Facts can be asserted and erased during a PRS run.

Production rules are defined as condition-action forms. A condition is a n-ary predicate whereby each predicate assumes a boolean value. At different times, these predicates can assume different boolean values. If all of the predicates of a production rules are "successful," then the production rule is said to have "fired" (i.e., the predicates have successfully matched and the action clause, an n-ary sequence of actions has been performed).

The interpreter matches condition clauses by comparing the predicates of a conditional clause to elements in the blackboard. If all of the predicates match some of the blackboard elements, then match has met with "Failure." As an example, if the blackboard contained the two elements

all tanks are weapons
and
all weapons are deadly

and if the rule set contained a rule

IF all tanks are weapons,
all weapons are deadly
THEN ASSERT (all tanks are deadly),

then the rule would fire because the predicates in the conditional clause match elements in the blackboard (e.g. "all tanks are weapons" matches "all tanks are weapons" and "all weapons are deadly" matches "all weapons are deadly"). This example has a conditional clause whose predicates are formed by literal phrases (i.e., there are no pattern variables). Extending the pattern match capability to handle general cases implies that we must introduce pattern variables. A pattern variable can match arbitrary values depending on the type of variable used. The PRS currently contains two types of pattern variables which are sufficient for the needs of this contract:

(1) Assigned variables

(2) Evaluated variables.

Assigned variables are preceeded by a "<" (e.g.,<VAR). The semantics of assigned variables are:

(1) Set the value of VAR to the next unmatched token in the blackboard element, and

(2) Return "SUCCESS."

Evaluated variables are preceeded by a "=" (e.g., =VAR). The semantics of evaluated variables are:

(1) Retrieve the value of VAR which must have been previously assigned by VAR, and

(2) Return "SUCCESS" if the next unmatched token in the blackboard element equals the retrieved value of VAR; otherwise, return "FAILURE."

Using the previous blackboard, the rule:

```
IF all tanks are <VAR,
all =VAR are deadly
THEN ASSERT (all tanks are deadly).
```

would succeed. VAR would be assigned the value "weapons" when "all tanks are weapons" matches "all tanks are <VAR." Then, "all weapons are deadly" would match "all =VAR are deadly" because VAR was previously assigned the value "weapons." Notice, that the second conditional clause uses =VAR instead of <VAR. This ensures that the variable VAR is the same value for both conditional clauses. Generalizing the last rule, it can be rewritten as:

```
IF all <subset are <superset,
all =superset are <characteristic
THEN ASSERT (all =subset and =characteristic).
```

This rule would successfully match either blackboard subset:

(1) All tanks are weapons, all weapons are deadly, or

(2) All concientious-objectors are pacifists, all pacifists are friendly.

As previously mentioned, action clauses are n-tuples which generally change the current PRS environment. The two commands which do not change the environment are:

(1) SEND, and

(2) RECEIVE.

SEND writes a message to the output device, and RECEIVE reads an input stream from the input device. (Note: RECEIVE may assign variables if the extended RECEIVE form is used.)

In PRS, new facts can be added to the blackboard by ASSERTING a fact. As an example, if the blackboard contained

all tanks are weapons
and
all weapons are deadly

and if the rule set contained the rule

```
IF all <subset are <superset,
all =superset are <characteristic
THEN ASSERT (all =subset are =characteristic);
```

then, on rule application, the rule would successfully fire and the blackboard would be updated. It would now contain

all tanks are deadly
and
all tanks are weapons
and
all weapons are deadly.

Notice that if this rule were applied again, it would fire and re-ASSERT that "all tanks are deadly." To prevent this from occurring, we can MARK blackgoard elements. This is the equivalent of deleting an item from the blackboard. Thus, if the action clause of the above rule were changed to

```
THEN MARK (all =subset are =superset),
MARK (all superset are =characteristic),
ASSERT (all =subset are =characteristic),
```

then the blackboard would essentially be equivalent to all tanks are deadly.

MARKed items are actually still in the blackboard; however, they are prevented from being matched. If it is desirable to pattern match against "marked" items, then they can be UNMARKed. To restore the two marked items in the above example, a rule can be written as follows:

```
IF  conditional clause which evaluates to "successful"
THEN UNMARK (all =subset are =superset
UNMARK (all =superset are =characteristic).
```

B. PRODUCTION RULES

In the IBC rule-based system, the rules are used primarily to direct the image processor. The IBC rules are designed to perform the following sequence of operations:

(1) Produce edge map

(2) Perform convention auto-cuer operations, which will segment possible targets

(3) Identify target segments from clutter

(4) Classify "unknown" targets using the model-based classifier

(5) Establish the target priority queue and define orientation window about the highest priority target

(6) Code target and orientation windows

(7) Send image to display unit.

A copy of the current IBC rules are included in Table 4-1.

Table 4-1. IBC Production Rules

```
R1:   If empty then assert (IBC STARTED),
        receive (<PICNAM),
        send (edgemap = PICNAM).

R2:   If edgemap complete then push (display edgemap = PICNAM 11),
        send (low = PICNAM),
        mark (edgemap complete).

R3:   If low complete then send (high = PICNAM),
        mark (low complete).

R4:   If jeep at <X <Y then assert (display jeep = PICNAM =X =Y),
        mark (jeep at =X =Y).

R5:   If truck at <X <Y then assert (display truck = PICNAM =X =Y),
        mark (truck at =X =Y).

R6:   If tank at <X <Y then assert (display tank = PICNAM =X =Y),
        mark (tank at =X =Y).

R7:   If APC at <X <Y then assert (display APC = PICNAM =X =Y),
        mark (PAC at =X =Y).

R8:   If unknown at <X <Y then send (region classification at
        = PICNAm =X =Y),
        mark (unknown at =X =Y).

R9:   If unidentified at <X <Y then mark (unidentified at +X =Y).

R10:  If clutter at <X <Y then mark (clutter at =X +Y).

R11:  If <T at <X <Y  then send (class = PICNAM =X =Y),
        mark (=T at =X =Y).

R12:  If IBC display start,
      display <OP = PICNAm <X <Y then send (display = OP = PICNAM
      =X =Y),
      mark (display = OP = PICNAM =X =Y).

R13:  If IBC display start then send (display Olsen = PICNAM 11),
      mark (IBC display start).

R14:  If Olsen display complete then receive ( PICNAM),
      send (edgemap = PICNAM),
      mark (Olsen display complete).
```

On the initial start-up of the IBC system, the blackboard is empty. This causes rule 1 to fire, asserting "IBC STARTED" and sends the command to the image interpreter to produce the edge map. When the edge map is complete, the image interpreter signals the controller by asserting 'EDGE MAP COMPLETE.' The controller then matches and executes rule 2, by starting the low-level segmentation and interest-point components of the auto-cuer. When the image interpreter is finished, the controller is notified by the assertion 'LOW COMPLETE.' Rule 3 is fired next, which causes the high-level auto-cuer segmentation to begin. The result of the high-level segmentation is a list of all possible target areas. An example of such a list returned to the knwoledge-based controller is shown in Figure 4-2. The next rule to be matched by the controller is rule 11, which begins the target classification procedure. During this phase, t he image interpreter uses a binary decision tree to distinguish targets from clutter and also, if possible, to classify the targets. If a setment can neither be discarded as clutter nor identified as a defined target type, it will be labeled an "unknown" target. For those targets that are classified, the target type together with the upper left corner coordinates are written to the blackboard. If a segment is identified as clutter, then the word "clutter" and the window coordinates are entered on the blackboard. Similarly, unknown targets are also asserted on the blackboard. Figure 4- 2 is an example of the target list on the blackboard after cuer low-level processing.

Once again, the control is returned to the knowledge-based system. The IBC controller will fire a combination of rules 4 through 10. Notice

```
RULE

R1. IF EMPTY
    THEN PUSH (IBC STARTED).
    RECEIVE (   PIC NAM).
    SEND (EDGEMAP • PIC NAM)

R2: IF EDGEMAP COMPLETE
    THEN PUSH (DISPLAY EDGEMAP • PICNAM 1 1).
    SEND (LOW • PICNAM).
    MARK (EDGEMAP COMPLETE).

R3: IF LOW COMPLETE
    THEN SEND (HIGH • PIC NAM).
    MARK (LOW COMPLETE).
```

```
           BLACKBOARD

      IBC STARTED
MARKED EDGEMAP COMPLETE
MARKED LOW COMPLETE
     T1    AT    69     5
     T2    AT   238     1
     T3    AT   160     1
     T4    AT   172     1
     T5    AT    53    18
     T6    AT    18    39
     T7    AT   135    60
     T8    AT     3    46
     T9    AT   158    58
     T10   AT   112    84
     T11   AT   118    97
     T12   AT    42   103
     T13   AT   187   102
     T14   AT    11   118
     T15   AT   204   160
     T16   AT   121   176
     T17   AT   206   209
```

AUGUST 1979

Figure 4-2. Example of IBC rules and resultant blackboard after cuer low-level processing.

that rule 10 simply marks a clutter window so that no furhter action need be considered on that segment. Those target areas known to be an APC, tank, truck, or jeep are ready for the display processor. However, for each "unknown" target area, a command is setn to the image interpreter to perform the model-based classification. The model-based classifier currently classifies an input target window as either a truck/jeep or as "unidentified." This information is added to the blackboard. Known targets are then asserted for display, and "unidentified" targets are simply marked for no further processing.

The knowledge-based controller is now ready to build the display image. Rule 12 commands the image interpreter to prioritize the targets and to define the orientation window. Rule 13 then initiates the simulated coding and decoding for the display image. As the production rule matches are improved, rules 12 and 13 will be replaced by rules to perform prioritization within the knowledge-based controller. Once the IBC display processing is complete, the controller will perform a match using rule 14 and thereby restart the entire IBC system at rule 2.

C. SYSTEM OPERATION

1. Priority Evaluation

Our current knowledge-based controller, limited us to simple predicate/action rules without the advantage of expressions. As a result, the necessary comparisons for target prioritization had to be moved down one level into the image interpreter. Prioritization is performed by a stand-alone program which stores all information on an intermediate disk file. The file consists of a 128-word block which contains the information to be given to the IBC display processor. A summary of the display block is given in Figure 4-3.

The eight-word window defintiion block contains information specifice to the edge map, orientation, and target windows. The name contains: (1) the disk file name containing the binary edge map for the edge map block. (2) 'ORIENTATN' for orientation window, (3) a blank for target windows. The X-start and Y-start coordinates reference the upper left corner of the window. For the edge map, this will be set to (1,1). The X-dimension defines the number of columns, and the Y-dimension defines the number of rows. The window type contains:

(1) 'EDGE' if edge map window block

(2) 'ORIEN' if orientation window block

(3) Target type if target window block (such as 'APC,' 'TANK,' 'JEEP," 'TRUCH').

The header contains miscellaneous information necessary for IBC display image generation. The noise flag is zero if no noise is introduced, otherwise a noise level representing the bit error rate is specified. If the gray flag is true, then the edge map background is set to the average gray level intensity of the original image; otherwise, a white background is used. The number of targets must be an integer between 0 and 10. The disk file name containing the original picture is used to calculate the average background intensity and also to extract the orientation and target windows. The target priority is defined by a vector in words 6 through 11 in the header block. The firs target type in the vector is the lowest priority; a maximum of six different target

9791-2

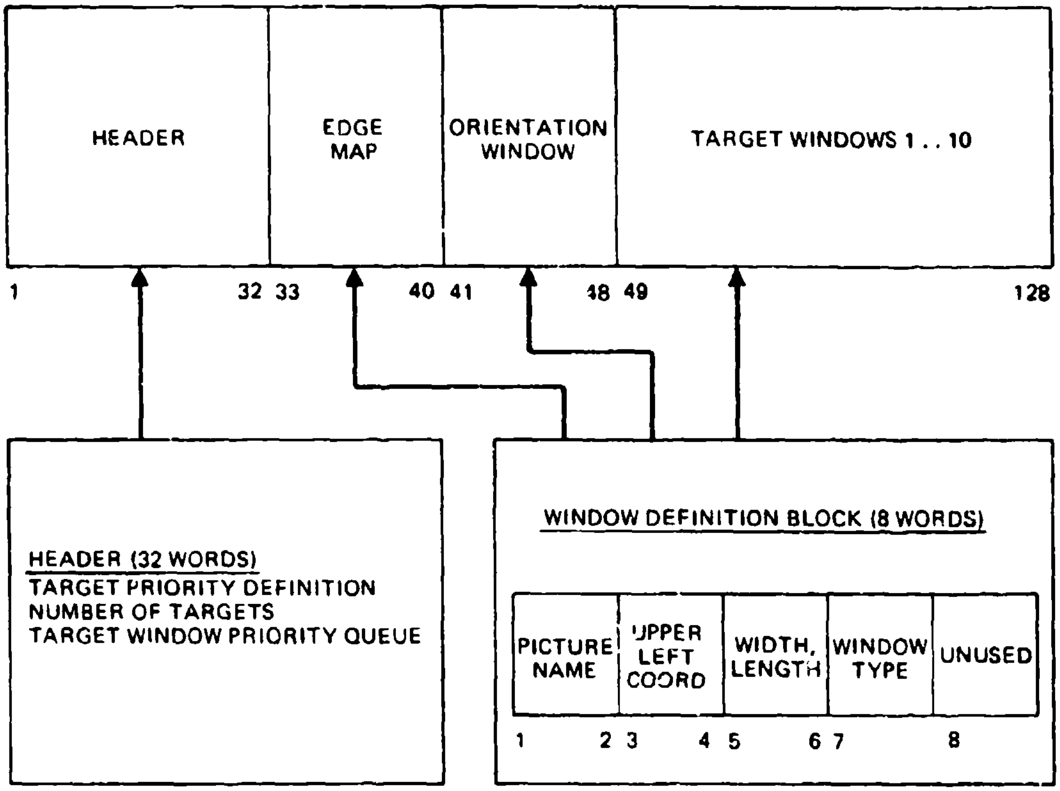


Figure 4-3. IBC display processor information block.

types may be prioritized by the operator. The priority vector for the classified targets in this picture in words 12 through 21. Figure 4-4 is an example of an IBC display processor information block. Notice that the target window priority vector consists of pointers to the first word of the corresponding window definition block.

Updating the IBC display with a new target window requires:

- Defining the target window block at the next available 8-word block
- Updating the target window block priority vector in the header (words 12 through 21) based on the defined priority vector (words 6 through 11).

If all window definition blocks have been assigned, then the lowest priority pointer in the target window priority vector will contain the area in which new target information will be stored. The new target type will be compared to the existing target types in the order of the target window priority vector (words 12 through 21) and in accordance with the target priority definition vector (words 6 through 11 in the header). The new target type is inserted before the first existing target of equal or lesser priority. If the new target is now the highest-priority target area in the image, a new orientation window is defined in the oreintation window definition block (words 41 through 48). Figure 4-5 is an update of the IBC display processor block in Figure 4-4. The new target APC has displaced the priority APC so that the target window priority vector was updated, and a new orientation window was defined.

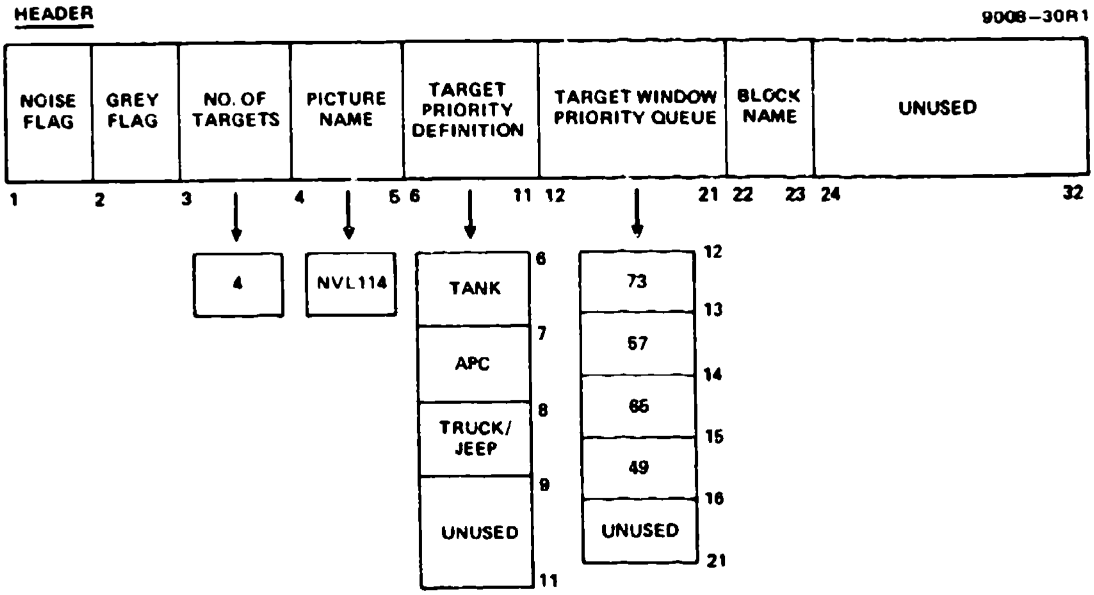


Figure 4-4. Example of IBC display processor information block.

9791-3

EDGE MAP

NVL114	1, 1	256, 250	EDGE	
33 34	35 36	37 38	39	40

ORIENTATION

NVL114	111, 83	123, 125	ORIENT	
41 42	43 44	45 46	47	48

TARGET WINDOWS

NVL114	122, 3	32, 32	TRUCK/ JEEP	
49 50	51 52	53 54	55	56

NVL114	159, 101	32, 32	APC	
57 58	59 60	61 62	63	64

NVL114	50, 127	32, 32	TRUCK/ JEEP	
65 66	67 68	69 70	71	72

NVL114	159, 131	32, 32	APC	
73 74	75 76	77 78	79	80

Figure 4-4. Continued.

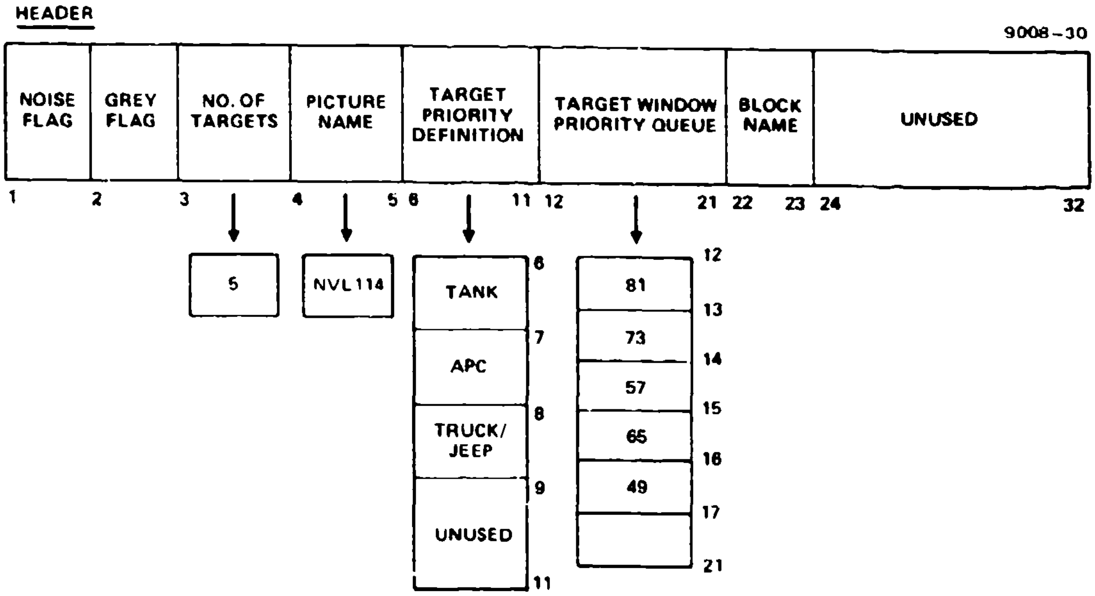


Figure 4-5. Example of updated 1BC display processor information block.

9791-4

EDGE MAP

NVL114	1, 1	256, 250	EDGE	
33 34	35 36	37 38	39 40	

ORIENTATION

NVL114	100, 98	128, 128	ORIENT	
41 42	43 44	45 46	47 48	

TARGET WINDOWS

NVL114	122, 3	32, 32	TRUCK/ JEEP	
49 50	51 52	53 54	55 56	

NVL114	159, 101	32, 32	APC	
57 58	59 60	61 62	63 64	

NVL114	50, 127	32, 32	TRUCK/ JEEP	
65 66	67 68	69 70	71 72	

NVL114	159, 131	32, 32	APC	
73 74	75 76	77 78	79 80	

NVL114	148, 146	32, 32	APC	
81 82	83 84	85 86	87 88	

Figure 4-5. Continued.

2. Classification

After the segmented objects have all been characterized by the statistical auto-cuer feature vectors, they are ready for use in the high-level classifier. The function of the classifier is to perform clutter versus target identification and recognition. A binary decision tree has been implemented to perform the classification. The binary decision tree format was chosen because it is easy to implement in a knowledge-based system. A tree structure easily divides into classes of target and clutter. Figure 4-6 shows a binary tree with the terminating nodes being either some target type or clutter. To classify a target and eliminate clutter segments based on statistically generated feature vectors, each branch of the tree is actually a numerical decision tree. An example of such a numerical decision tree is given in Figure 4-7. The auto-cuer features found for our test images were to be most reliable for distinguishing targets from clutter were

- Area
- Intensity threshold used to segment the object from the background
- The number of points within the interior of the segmented object that intersect with the interest window boundary.

For instance, a road segment may be greater in area than an APC. In our data base, the roads were always very bright in comparison with the background and therefore possessed a high threshold value. But, most importantly, as the road corssed through the segmentation window, many points coincided with the window boundary, whereas most targets were contained entirely within the interest window.

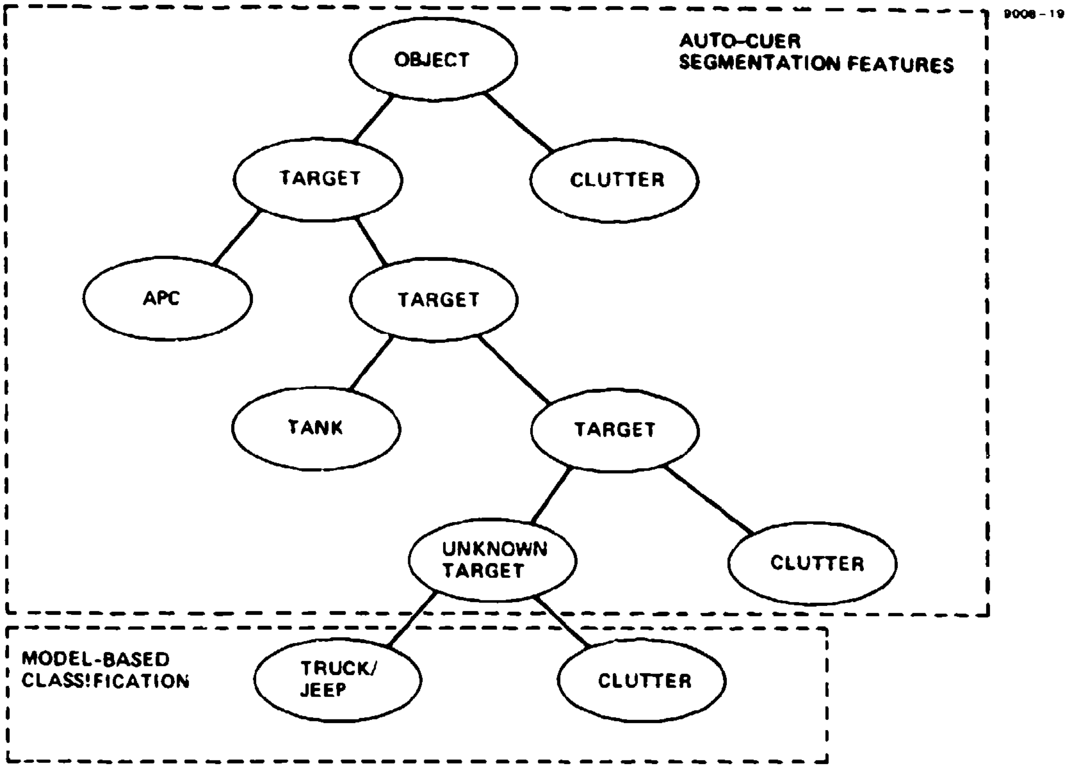


Figure 4-6. Binary decision tree.

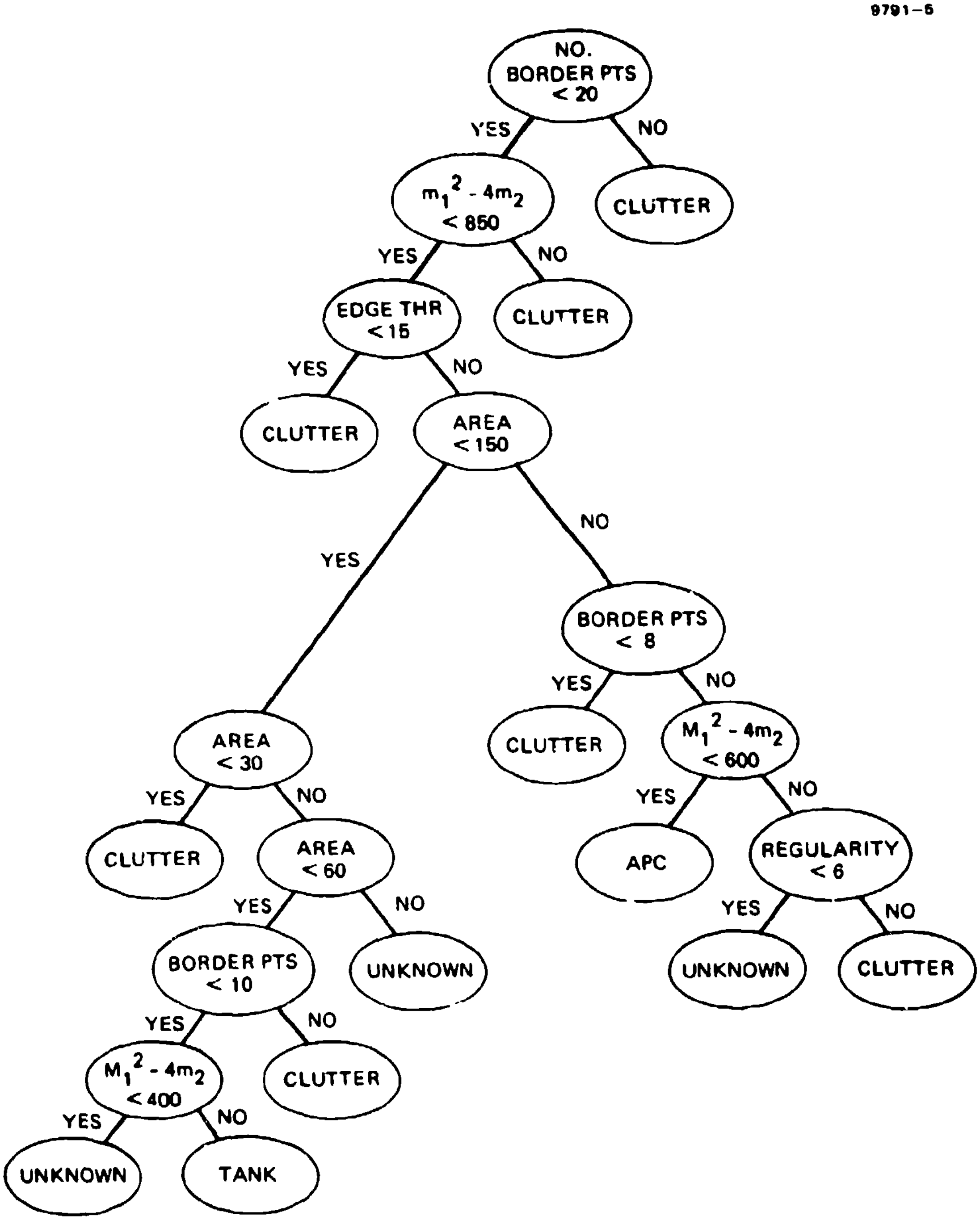


Figure 4-7. A numerical decision tree to classify an APC or tank based on auto-cuer segmentation features.

To identify and classify target windows, additional features are required. The visible band imagery produced some new problems in target classification. Areas of high reflectivity and the effects of shadowing changed with sun angle, which produced very different segmentation results. However, the first two invariant central-moment transforms; the length, width, and aspect (length/width) derived from the calculated moments; and the area have all worked well with our data base.

As stated earlier, the binary classification tree was selected because it fit easily into the knowledge-based system. Our current knowledge-based controller has not yet been extended to accept prediate expressions. Together with the binary classification tree, this has been programmed as part of the image interpreter. The code is designed to be modular so that, as the knowledge-based system develops, the production rules will be able to replace the current decision tree modules. An example of a set of classification rules are included in Figure 4-8.

9791-6

```
R1  IF    TARGET = n AND
          ((20 ≤ BORDER PTS)
          OR $(850 \le m_1^2 - 4m_2)$
          OR (EDGE THRESHOLD < 15)
          OR (AREA < 30)                  )
    THEN  SEND (CLUTTER AT TARGET = n),
          MARK (TARGET = n).

R2: IF    TARGET = n AND
          (8 ≤ BORDER PTS < 20) AND
          $((m_1^2) < 600)$ AND
          (15 ≤ EDGE THRESHOLD) AND
          (150 ≤ AREA)
    THEN  SEND (APC AT TARGET = n),
          MARK (TARGET = n).

R3: IF    TARGET = n AND
          (BORDER PTS < 10) AND
          $(400 \le (m_1^2 - 4m_2) < 850)$ AND
          (15 ≤ EDGE THRESHOLD) AND
          (30 ≤ AREA < 60)
    THEN  SEND (TANK AT TARGET = n),
          MARK (TARGET = n).
```

Figure 4-8. Sample set of classification rules.

SECTION 5

CODING AND CONSTRUCTION

The source and channel coding schemes for the IBC system have been computer simulated. A block diagram showing a general model of a video transmission system is shown in Figure 5-1. From a communications point of view, the source and channel encoder are the two highest-level components of the remote IBC terminal. Note that, from this viewpoint, the source encoder contains all of the sophisticated image-interpretation and priority-control functions of the IBC system. The function of the source encoder is to represent the image as efficiently as possible (i.e., with the fewest bits) while the channel encoder intentionally adds redundancy to make the information less subject to errors (e.g., jamming).

Three classes of data are extracted by the IBC system for transmission on the downlink: the edge map video, the conventionally compressed sub-image windows, and the symbolic descriptors. A typical channel resource allocation a 1,000:1 compression ratio is shown above in Table 1-1. The preliminary edge map source encoding method is a 1-bit scanning at 1/4 resolution. This requires 16.4 kbit/sec at a 1 fps update. A 32 x 32 pixel sub-image, centered on the highest priority target, is encoded at 1.5 bits per pixel (bpp) and updated at 7-1/2 fps. A 128 x 128 pixel orientation (or reference) window is transmitted at 1 bpp and 1 fps to provide orientation information to the ground station operator. The channel resource allocation also provides for transmission of subwindows of 32 x 32 pixel size for secondary targets. These are updated

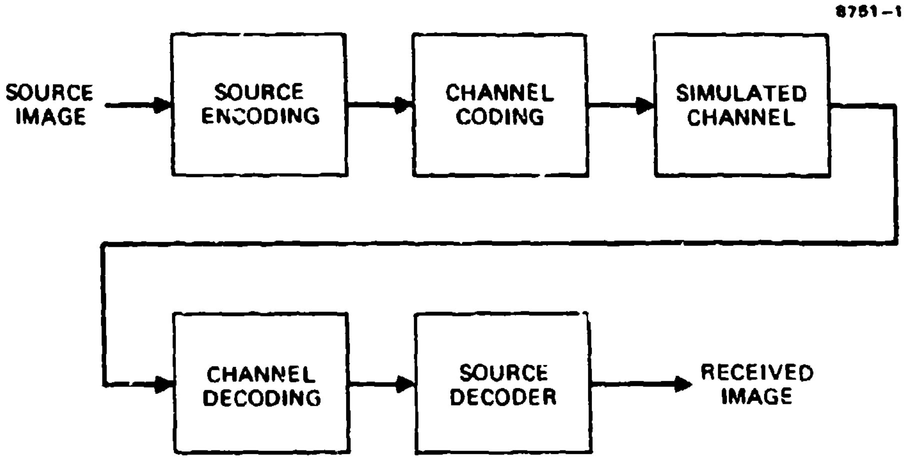
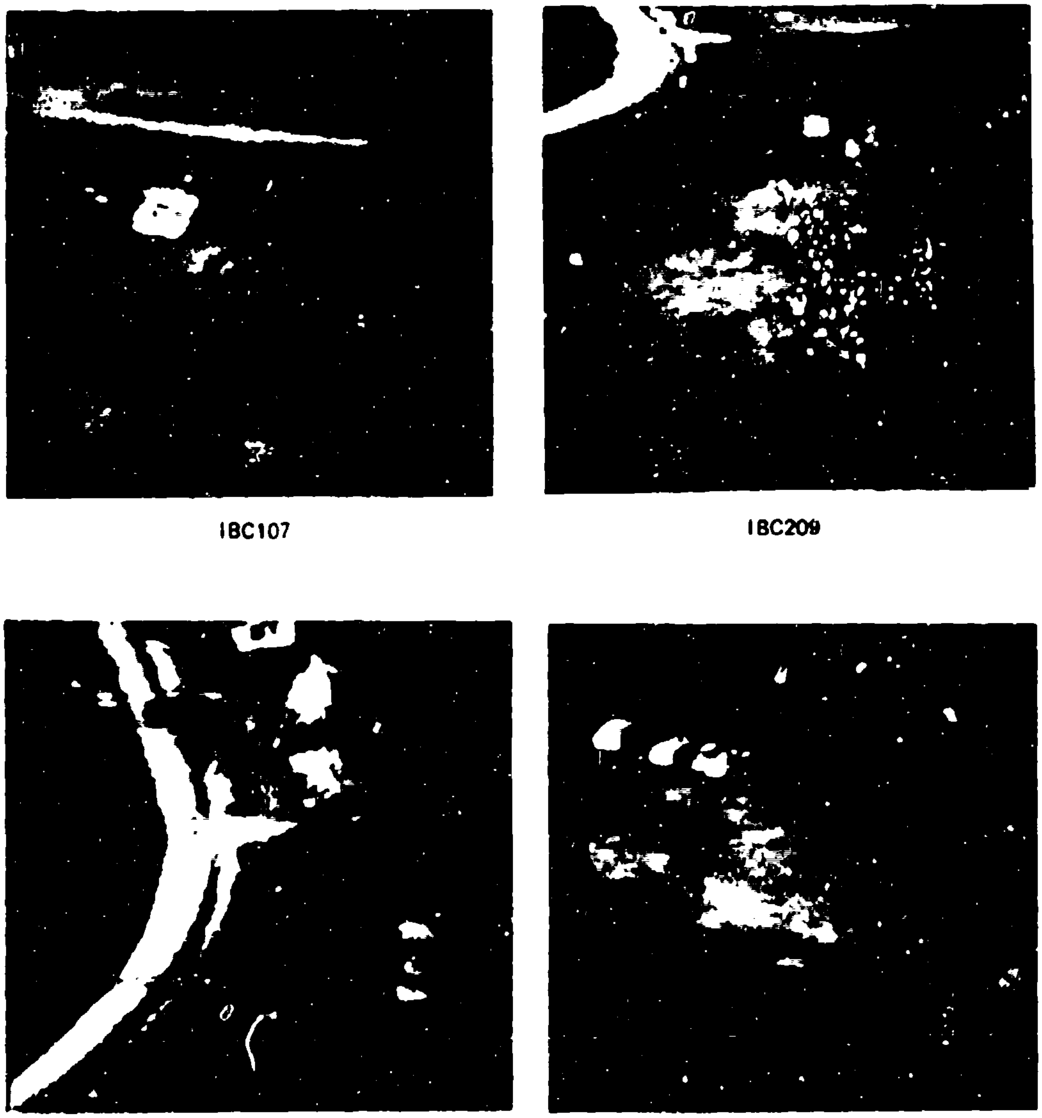


Figure 5-1. General model of video transmission system.

at 1 fps and would be coded at 1 to 2 bpp, depending on the number of secondary targets to be transmitted. The symbolic descriptors are merely digital messages of approximately 48 bits in length. Hence, performance of the communications system in transmitting these messages can be readily evaluated (e.g., bit error rate curve) and was not simulated during the first quarter. Computer simulation results of source and channel coding of the edge map and conventionally compressed image subwindows, as well as the reconstructed IBC images, are discussed below.

1. Edge Map Coding

Four images from the IBC data base are shown in Figure 5-2, with the corresponding full resolution (256 x 250) edge maps shown in Figure 5-3. Using the raster transmission format, the edge maps are seen in Figure 5-4 to be rather robust. Even at BER = 10^{-1}, the edge map is still very usable. However, a typical modem operating point is more like BER = 10^{-3}. A ground-restoration technique called edge-map filtering was used to remove isolated point errors. The filtering algorithm removes any indicated edge point that does not have nearest neighbors. The performance of these filtering techniques can be readily estimated. At an error point in a black region of the edge map, the probability that none of the eight neighbors are also in error is $(1 - P_B)^8$, where P_B is the channel bit error rate. Substituting a channel BER = 10^{-2} gives the result that 92% of the channel errors should be removed by the filtering algorithm. The result of applying the filtering algorithm on the noise-corrupted edge maps of Figure 5-4 are shown in Figure 5-5. Note that at BER = 10^{-2}, 644 corrections were made out of 676 errors, which is consistent with the 92% error correction prediction. The channel capacity



IBC107

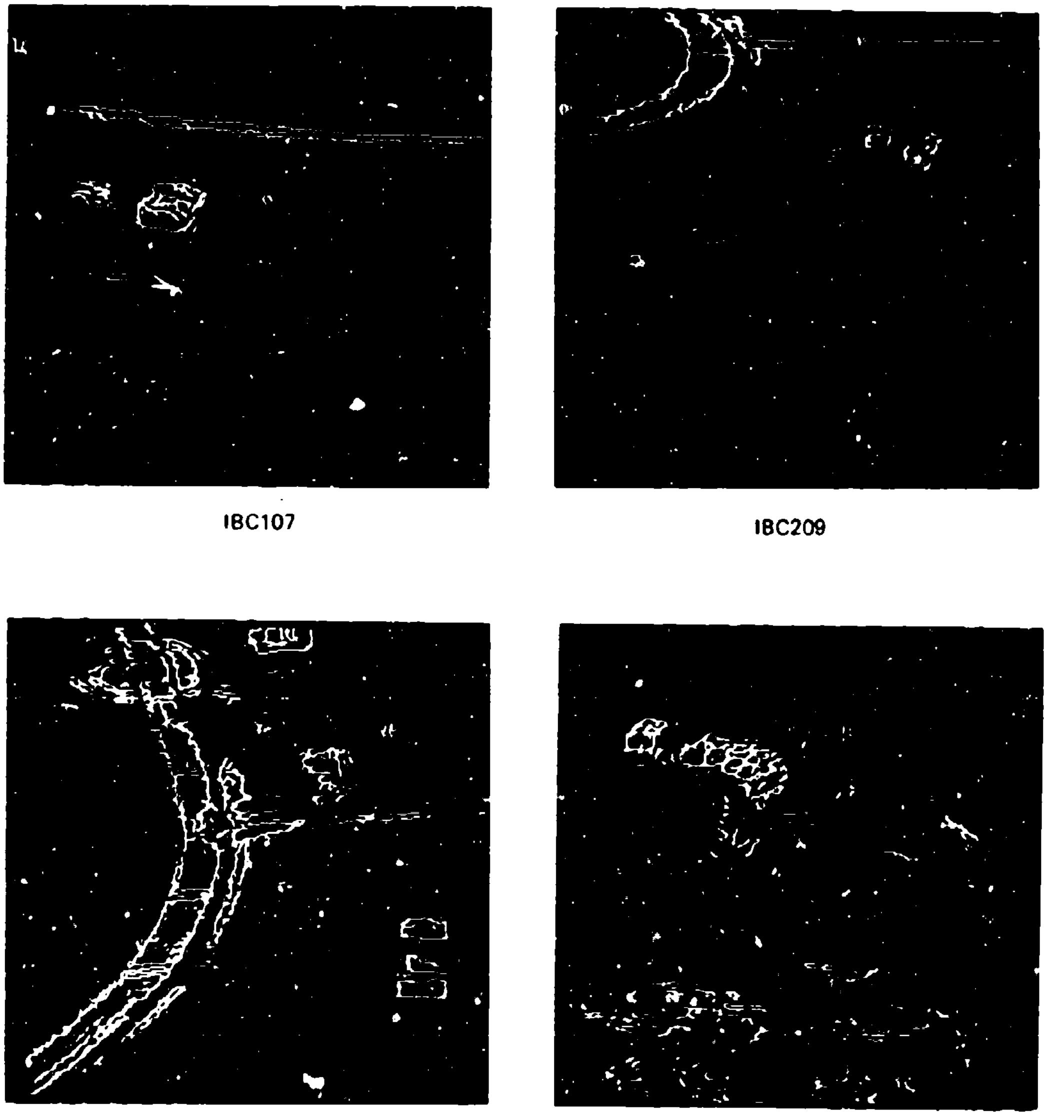
IBC209

IBC111

IBC201

Figure 5-2. Representative Images from IBC data base.

8761-12



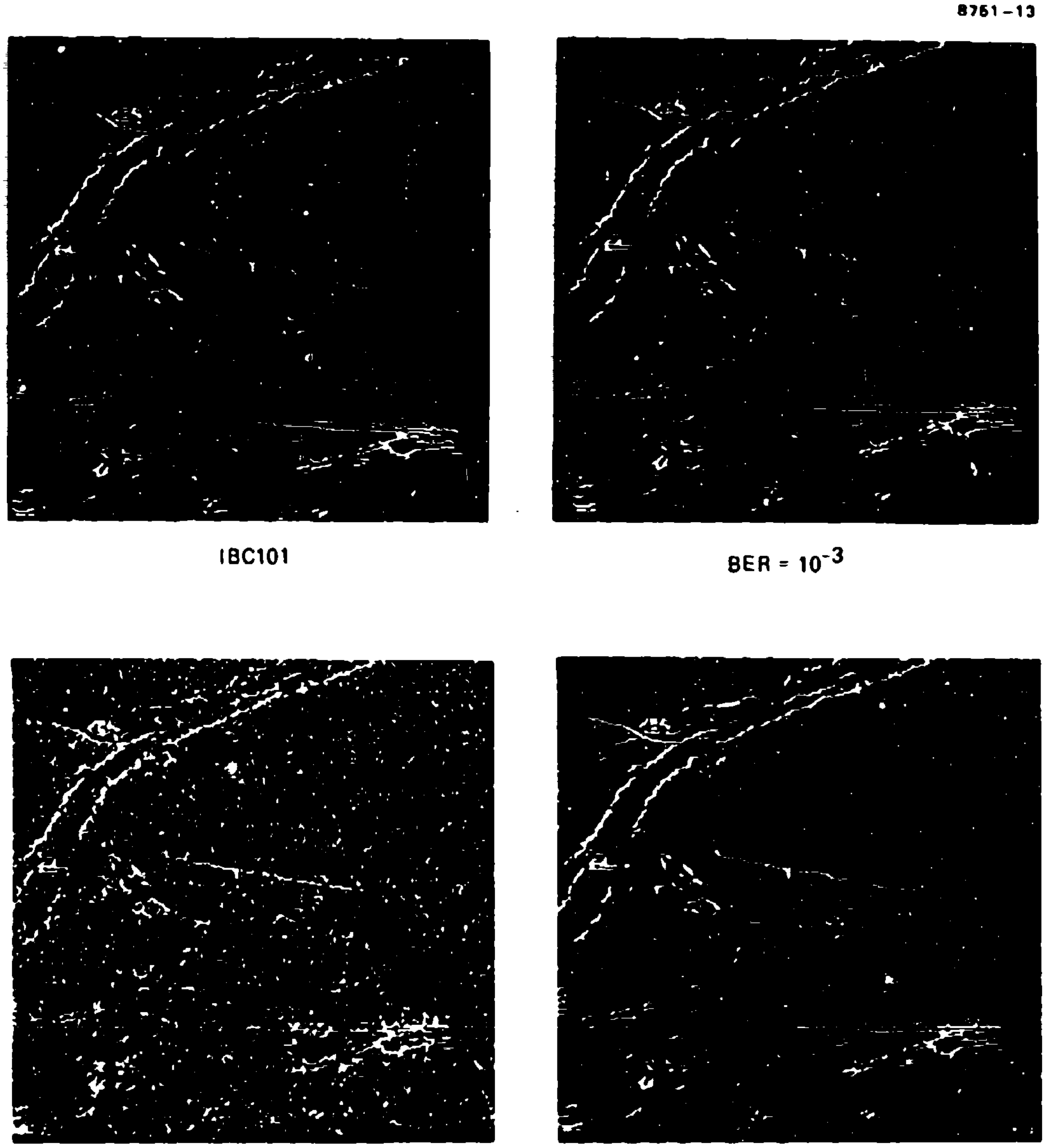
IBC107

IBC209

IBC111

IBC201

Figure 5-3. Edge maps (full resolution).



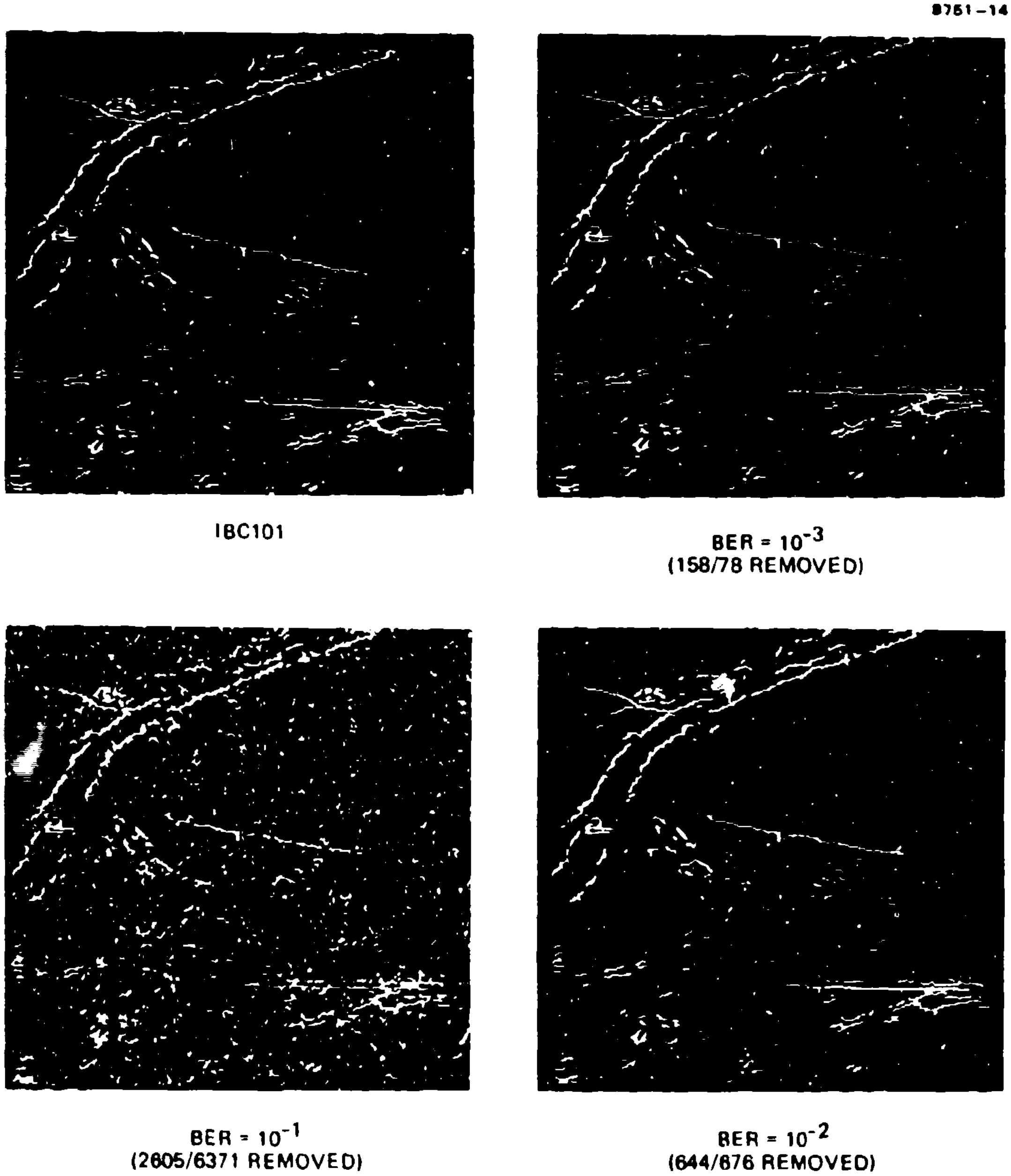
IBC101

BER = 10^{-3}

BER = 10^{-1}

BER = 10^{-2}

Figure 5-4. Edge coding.



IBC101

BER = 10^{-3}
(158/78 REMOVED)

BER = 10^{-1}
(2605/6371 REMOVED)

BER = 10^{-2}
(644/676 REMOVED)

Figure 5-5. Filtered edge map.

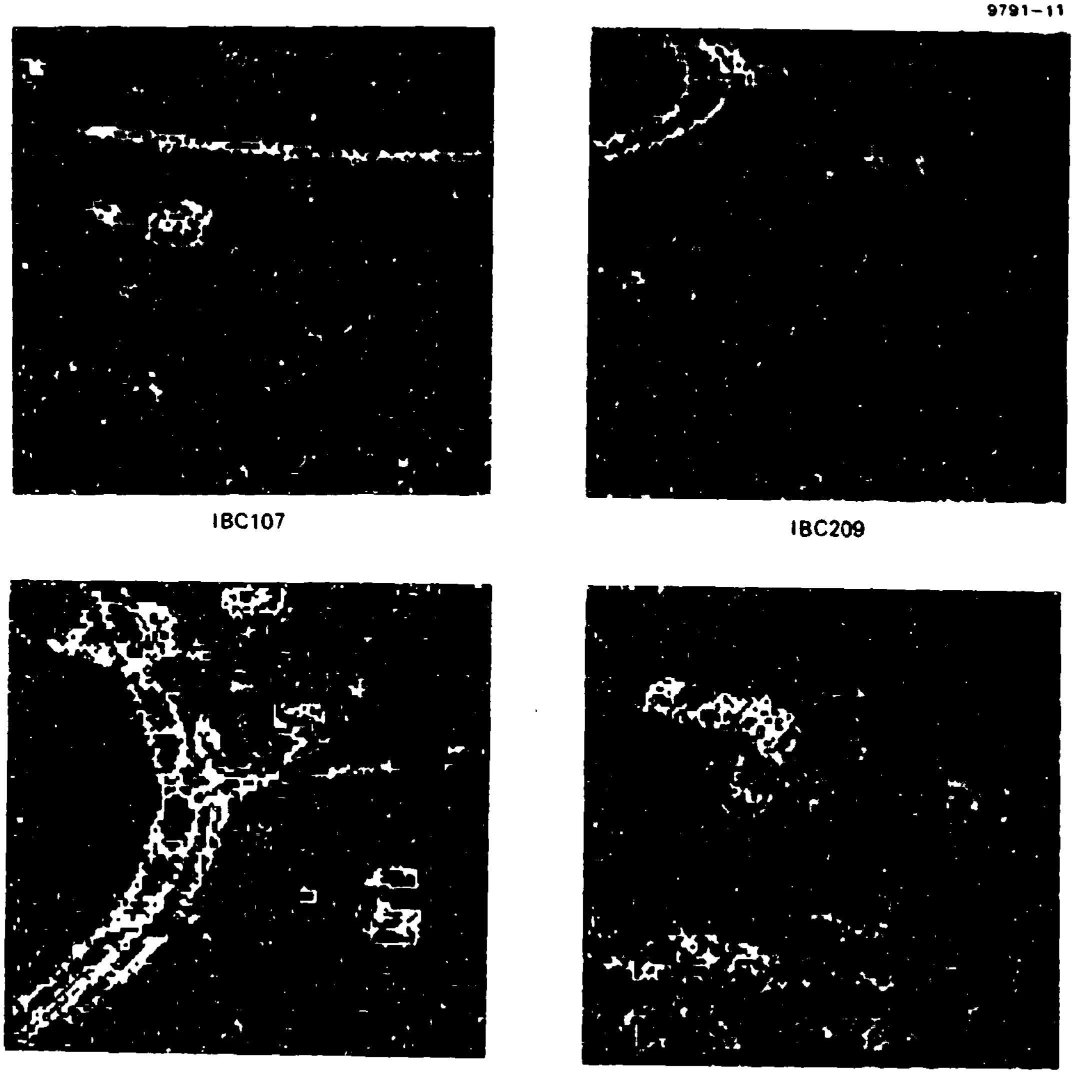
at 1,000:1 compression (i.e., 63 kbit/sec) is insufficient for transmission of the full resolution edge map. Therefore, the edge images were reduced in resolution by a factor of two in each dimension.

The final results of the edge map coding simulation are shown in Figure 5-6 for a channel BER = 10^{-2}. Note that these images have been processed as follows: (1) resolution reduction, (2) channel error simulation, (3) edge map filtering, and (4) expansion to a 256 x 250 resolution.

Two classes of conventional image coding techniques — predictive and transform encoding — were studied for the transmission of subwindow video. From each class, a specific technique was selected as representative. These techniques, differential pulse code modulation (DPCM) and one-dimensional Walsh-Haramond transform, were used to encode the IBC data base. The results are descussed below.

a. DPCM Coding

DPCM, a simple technique, is frequently used for image coding. Dynamic range by reduction and subsequent bandwidth reduction are achieved by encoding the difference between a predicted pixel amplitude (based on previous samples) and the actual amplitude. The mean and variance on both a per pixel and a differential basis were computed on subject images from the NVL data base. The statistics for image IBC 101 are shown in Table 5-1. Note that differential encoding reduced the variance by more than a factor of 20. The computed inter-pixel correlation coefficients, used by the predictor, were typically very large for this data base. However, a large prediction coefficient caused channel errors to propagate extensively (Figure 5-7). Hence, a prediction



IBC107

IBC209

IBC111

IBC201

Figure 5-6. Edge coding (reduced resolution transmission). BER = 10^{-2}.

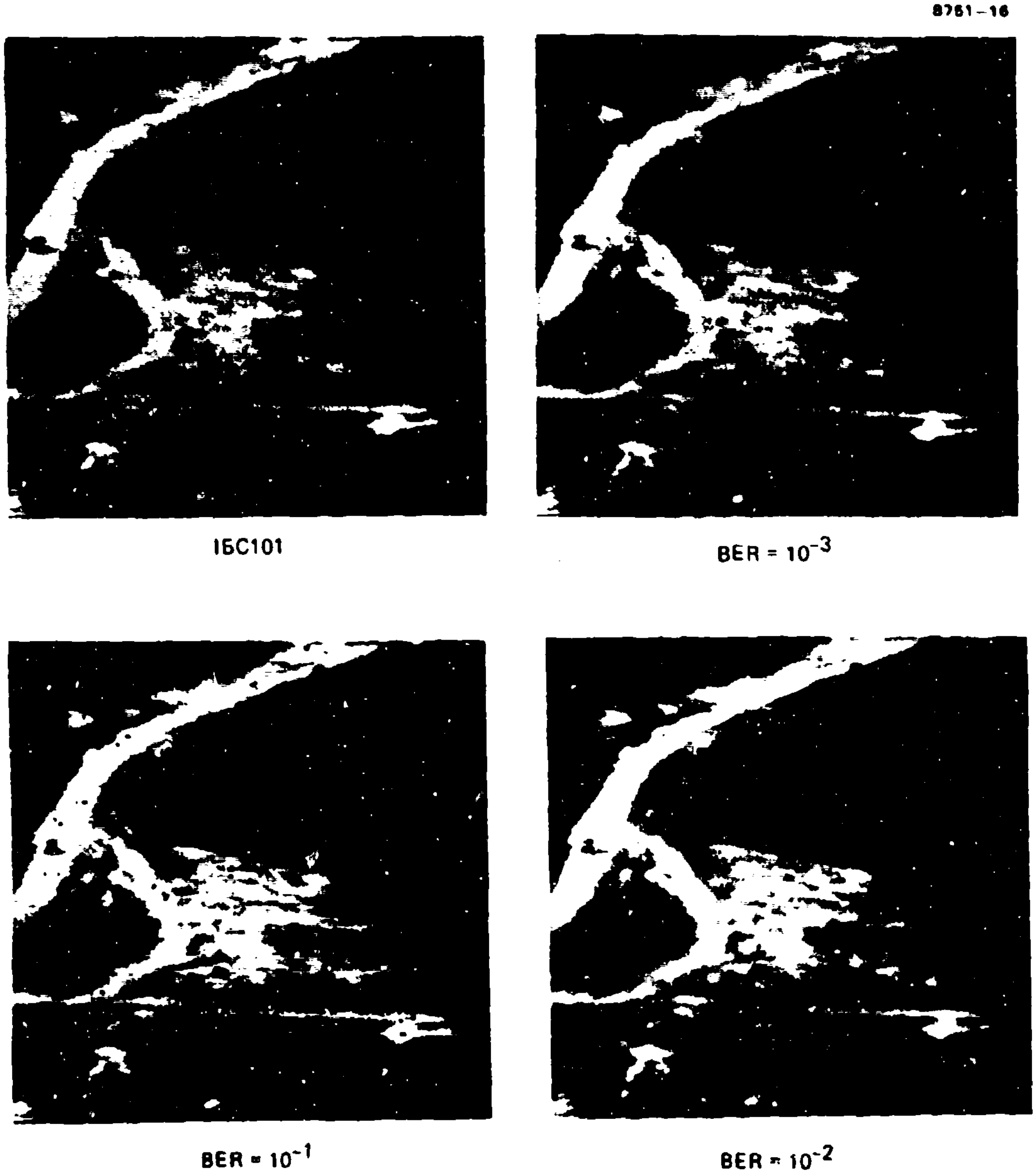


Figure 5-7. DPCM coding with channel noise. (ρ = 0.98, 2 bpp).

coefficient of only 0.9 was used in all subsequent DPCM coding simulations, the results of which are shown in Figure 5-8 for 2-bit DPCM. At BER = 10^{-1}, the image is very streaked due to error propagation.

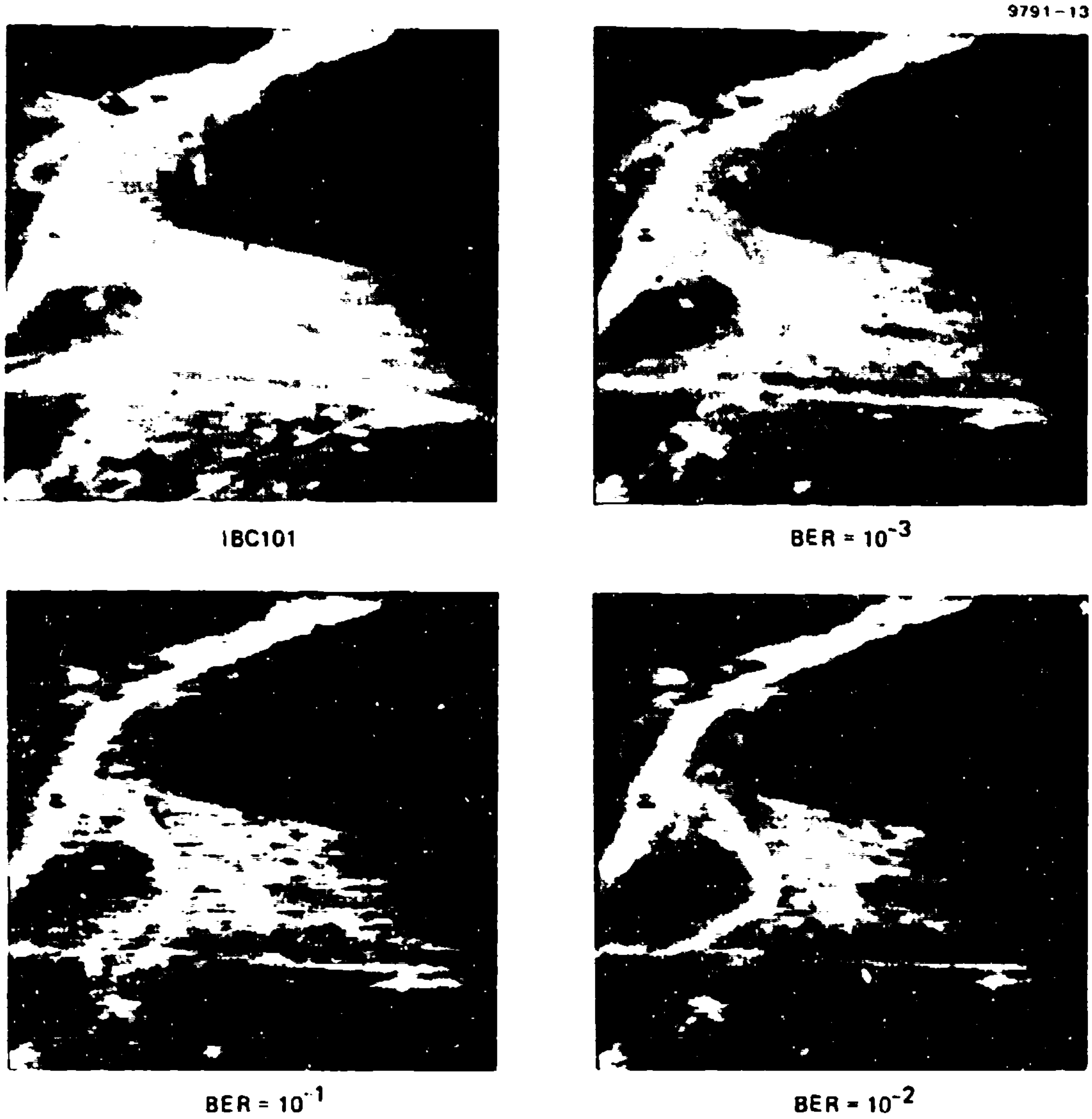
b. Hadamard Transform Coding

The Hadamard transform coding simulation is shown in the block diagram of Figure 5-9. The image (or sub-image) was first operated on by a one-third power memory-less nonlinearity and then undergoes a coordinate transformation via the one-dimensional, 16-point Walsh-Hadamard transform. Each transform coefficient is then individually quantized, depending on the component's importance in terms of energy contents and the human sensitivity to errors in that component. The quantized coefficient were then error correction encoded and corrupted with bit errors by the channel simulator. The decoding operation is essentially the reverse of the encoding operation.

The one-third power preprocessing mimics the nonlinear processing found in the human visual system (typically modeled as a logarithmic function) and is intended to reduce the human perception of the quantization errors.[17] The quantizers are designed as nonlinear instantatneoulsy companding quantizers in which a memory-less nonlinearity is performed

Table 5-1. DPCM Statistics for IBC 101

Mean (pixel)	116
Variance (pixel)	1841
Correlation coefficient	0.98
Mean (delta)	-0.1
Variance (delta)	76.8



IBC101

BER = 10^{-3}

BER = 10^{-1}

BER = 10^{-2}

Figure 5-8. DPCM coding with channel noise (P = 0.9, 2 bpp).

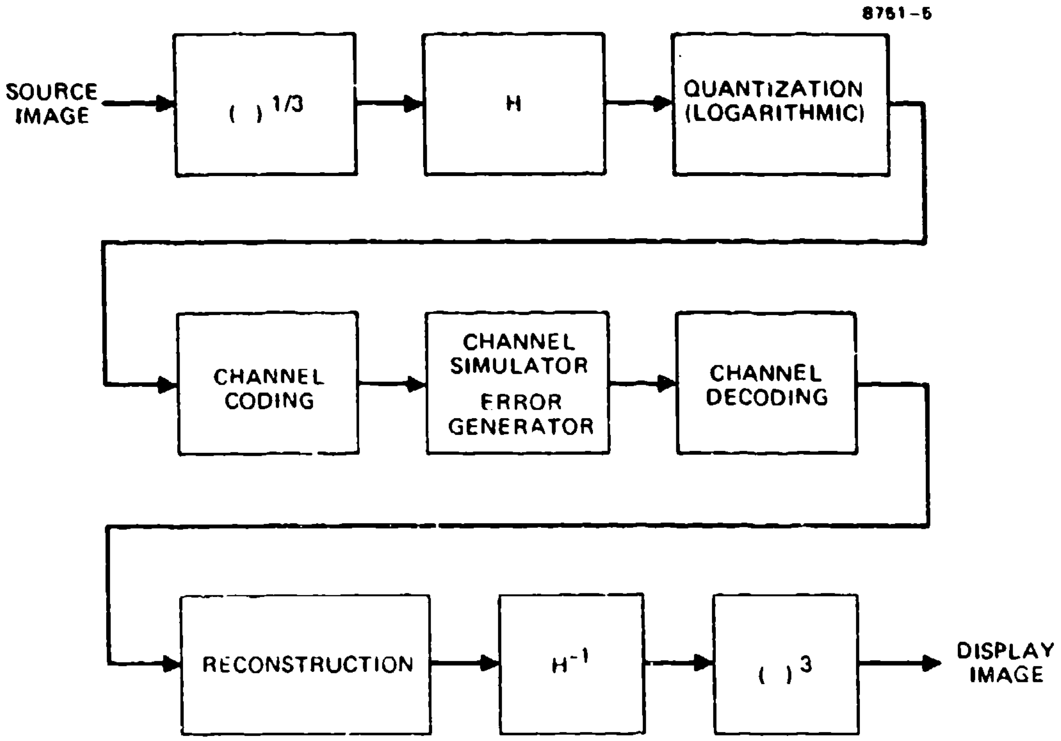
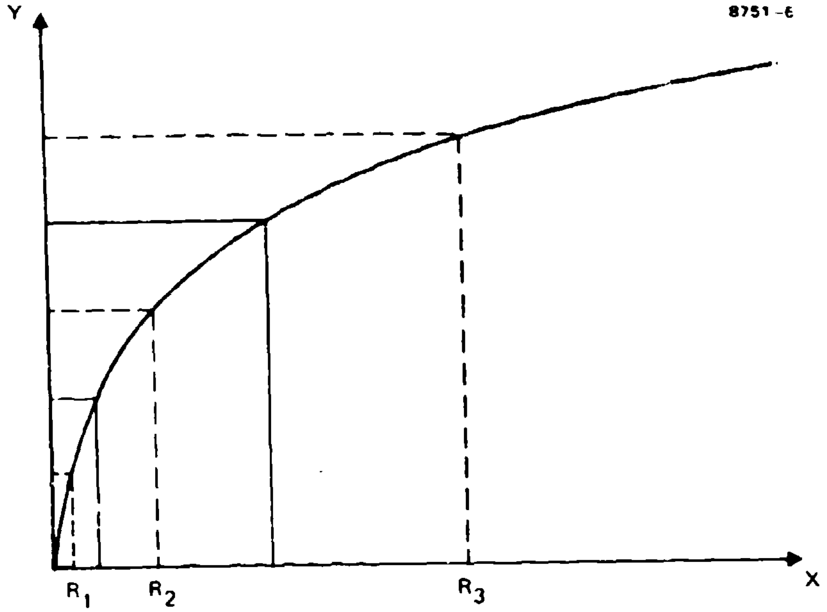


Figure 5-9. Block diagram of Hadamard transform coding simulation.

prior to uniform quantization. A logarithmic function is indicated to minimize the human perception of encoding errors.[17] A useful feature of these instantaneously companding quantizers is that, to more coarsely quantize a coefficient (i.e., greater bandwidth compression), bits are simply deleted from the quantizer output without changing the quantizer cut points. The logarithmic quantizer is diagrammed in Figure 5-10 with the cut points shown as solid lines and the reconstruction levels shown as dotted lines.

The transform domain image statistics (Table 5-2) are used to determine bit allocations among Hadamard coefficients (sequencies) and the dynamic range (spread) of the quantizer. The four columns in Table 5-2 are the Hadamard sequency number, sequency mean, sequency variance, and percentage of total picture variation. Note that there is little difference in the image variance distribution among transform components whether the one-third power preprocessing is used (Table 5-2a) or is not used (Table 5-2b). The Hadamard transform coefficient bit allocations for 2 bpp, 1.5 bpp, and 1 bpp are shown in Table 5-3. Note that these allocations account for human sensitivity to encoding errors and were not just derived from a rate-distortion theory and mean-squared-error computation based on the image statistics of Table 5-3 (the allocations are substantially different).

An alternate 1-bpp bit-assignment algorithm also shown in Table 5-3, which had been the standard Hughes algorithm, was established based on a sizable image data base and an experimentation with the Hughes real-time



FC℞.·ı OF NONLINEARITY f(X) = A · LOG (1. + B · X)

Figure 5-10. Logarithmic quantization.

Table 5-2. Hadamard Statistics (IBC101)

Sequency	Sequency Mean	Sequency Variance	Percent of Total Picture Variation
0	482.02475	24083.15600	0.90545
1	2.10875	1391.49840	0.05332
2	0.08025	353.83481	0.01389
3	1.33525	308.42336	0.01160
4	-0.21775	60.48983	0.00227
5	0.03925	77.48471	0.00291
6	-0.06425	79.43212	0.00299
7	0.62775	78.37916	0.00295
8	0.01925	17.71188	0.00067
9	-0.10675	16.64785	0.00063
10	0.05075	16.85267	0.00063
11	0.07175	16.62110	0.00062
12	-0.09825	20.33760	0.00076
13	0.01675	23.40997	0.00088
14	-0.01475	25.78803	0.00097
15	0.38725	47.99729	0.00185

(a) Linear Processing

Table 5-2. Continued.

Sequency Sequency	Sequency Mean	Sequency Variance	Percent of Total Picture Variation
0	19.63224	5.73112	0.84962
1	0.63497	0.55466	0.88223
2	0.00086	0.14132	0.02895
3	0.02163	0.12352	0.01831
4	-0.00867	0.82505	0.00371
5	0.00007	0.03159	0.00468
6	0.00135	0.03257	0.00486
7	0.01032	0.03243	0.00481
8	0.00002	0.00629	0.00123
9	-0.00189	0.00759	0.00113
10	0.00050	0.00762	0.00116
11	0.00106	0.00734	0.00109
12	-0.00116	0.00694	0.00133
13	0.00021	0.01014	0.00150
14	-0.00026	0.01121	0.00168
15	0.00585	0.01209	0.00179

(b) 1/3 Power Pre-Processing

Table 5-3. Hardemard Bit Assignments

	2 bpp	1.5 bpp	1 bpp	Alternate 1 bpp'
S_0	5	5	5	4
S_1	5	5	5	4
S_2 S_3	3	3	3	2
S_4 S_7	2	2	C	1
S_8 S_{15}	1	0	0	0
Total	32	24	16	16

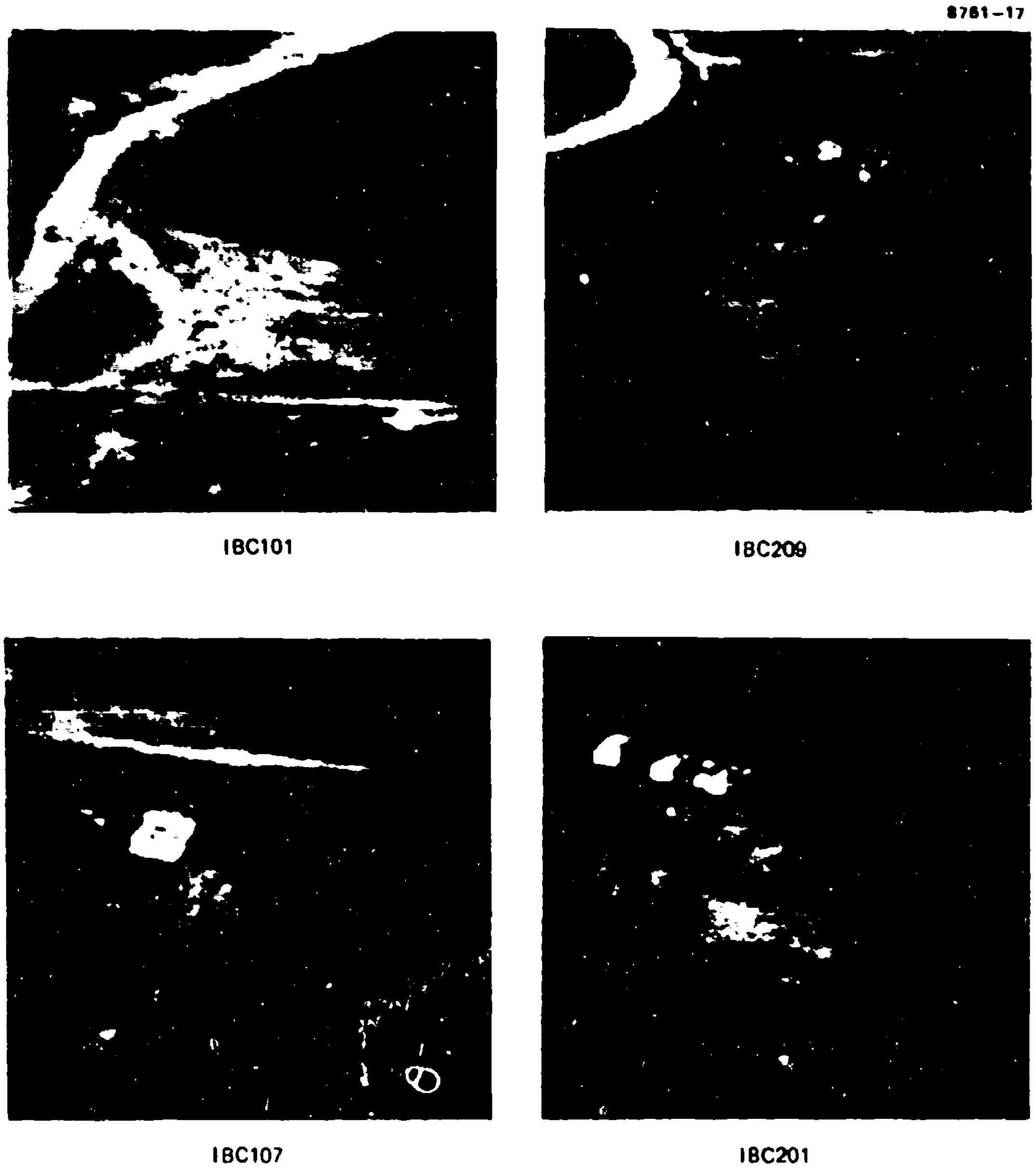
Hadarmard video processor. However, due to the very high transform domain energy compaction (∿90%) exhibited with the IBC data base, the first 1 bpp algorithm appears to be preferable and was used in the simulated IBC display results shown later in this section. The results of Walsh-Hadamard coding of the NVL data base are shown in Figures 5-11 to 5-14 for 2 bpp, 1.2 bpp, 1 bpp, and 1 bpp' respectively.

Note finally that there is very little difference between the 2 bpp and the 1.5 bpp images.

4. IBC Reconstruction

Figure 5-15 shows several versions of the simulated ground display at a data rate of 50 kbps, with Hadamard sub-image coding. These displays show the edge-map, the 128 x 128 orientation window, and two 32 x 32 priority target sub-images. We felt that using the average background gray scale as the non-white level in the edge maps would enhance the ground display. This enhancement technique is shown in the lower right with a background level of 127 out of a range of 0 to 255. If the gray scale had actually been chosen as the average background level, the display would have been even more pleasant. In the upper right, the primary and secondary target windows have been highlighted by adding bright and dark borders, respectively.

Figure 5-16 shows a reconstructed image that has been completely processed by the computer-simulated IBC system. The latest auto-cuer results indicated 3 detections, with one false alarms. Each indicated target was coded with a 32 x 32 pixel subwindow at 2 bpp. The 128 x 128



IBC101

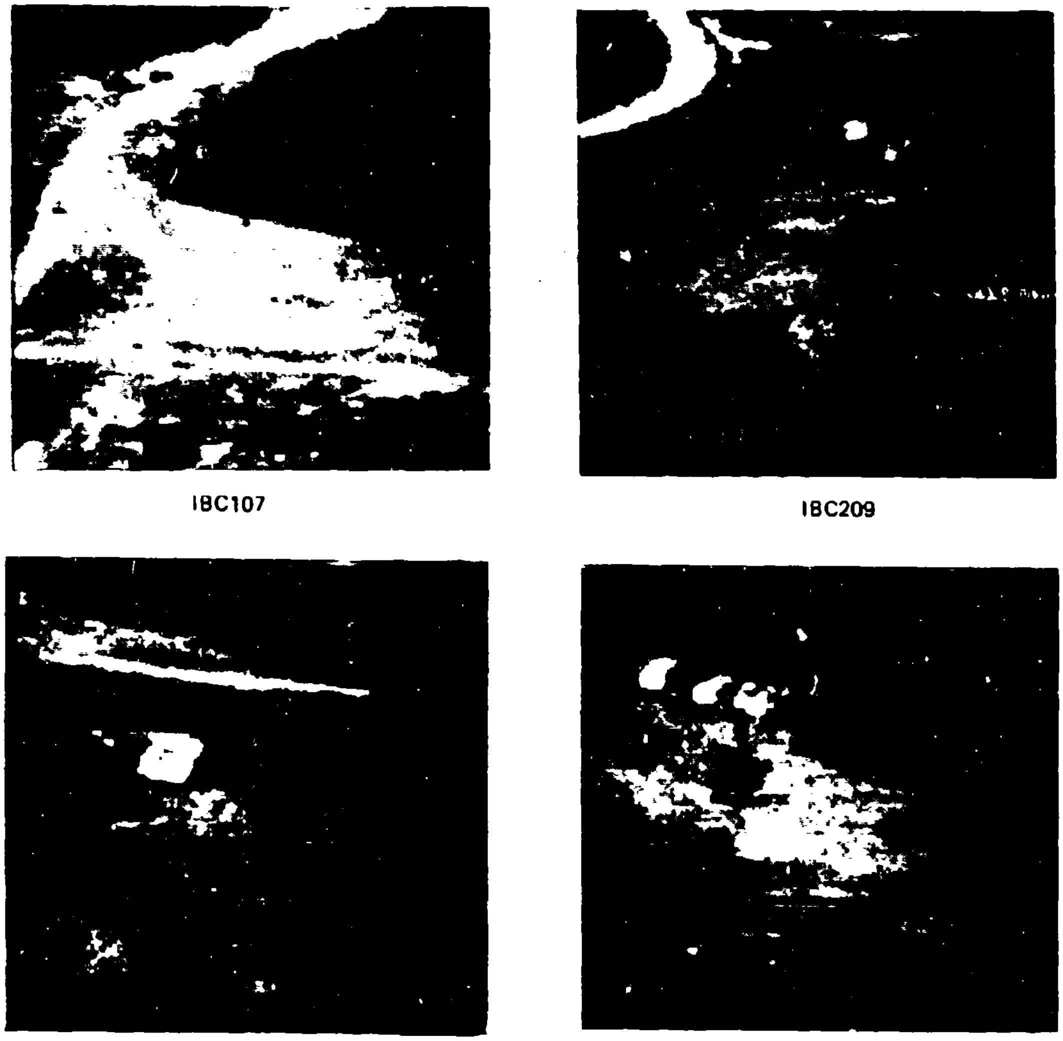
IBC209

IBC107

IBC201

Figure 5-11. 2-bit Hadamard coding.
BER - 10^{-2}.

9791-15



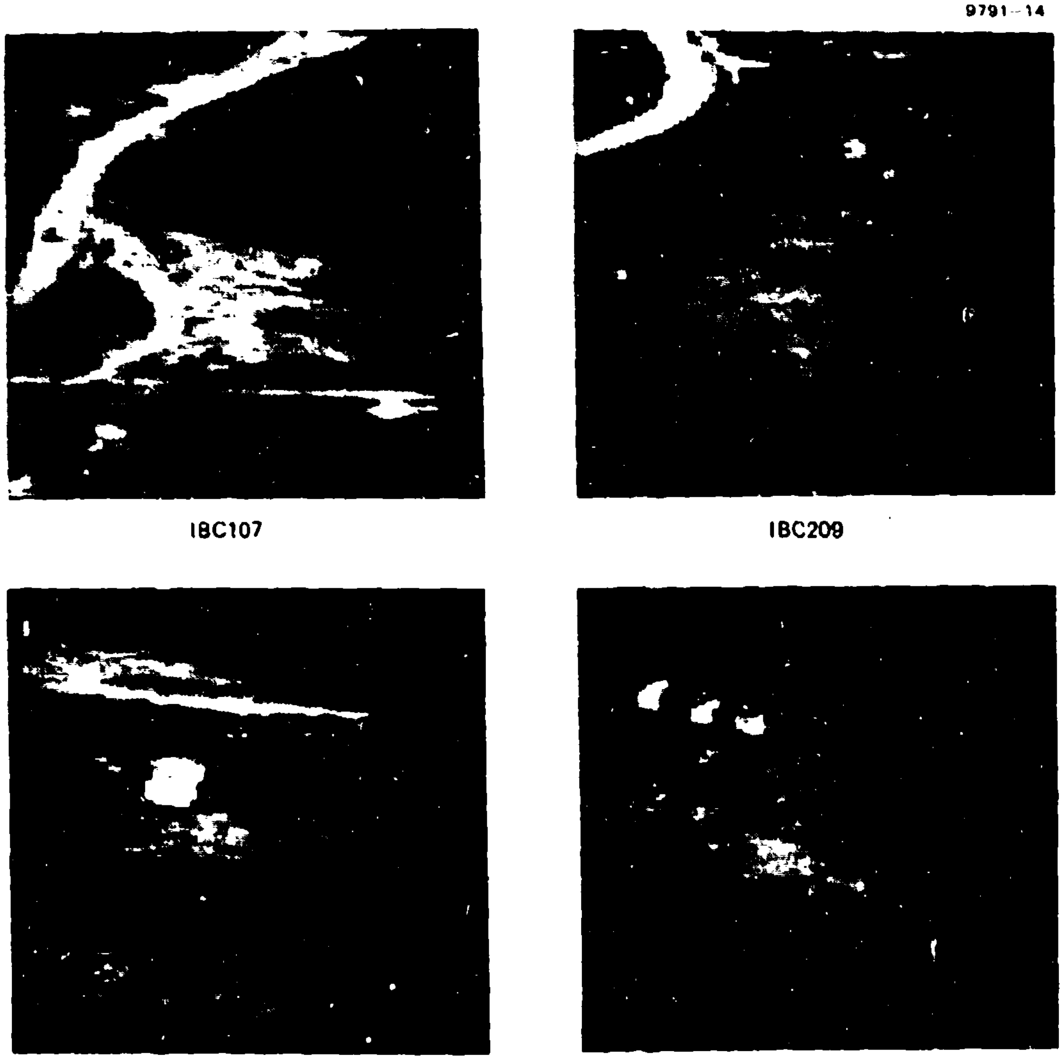
IBC107

IBC209

IBC101

IBC201

Figure 5-12. 1.5-bit Hadarmard coding BER = 10^{-2}.



9791-14

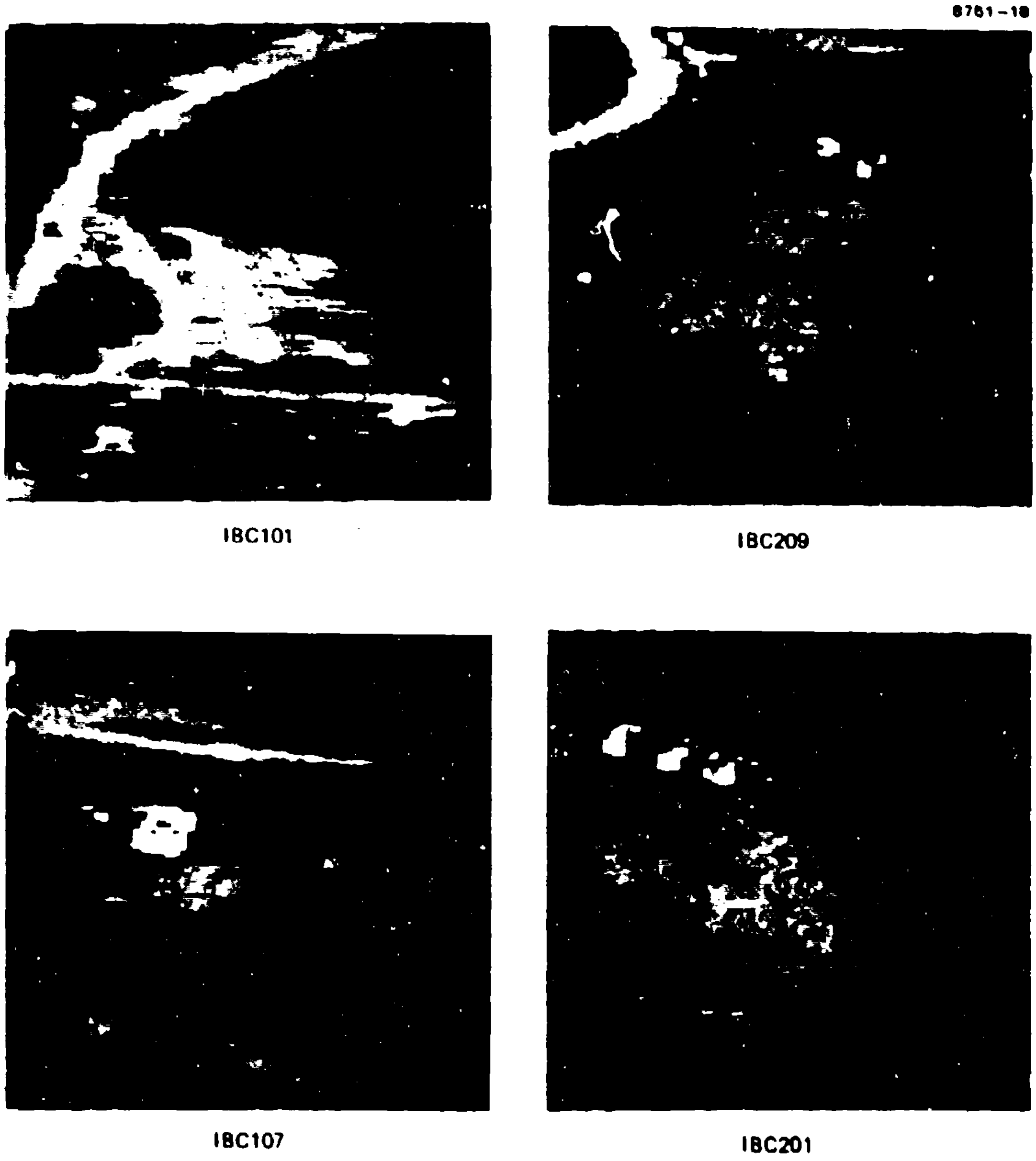
IBC107

IBC209

IBC101

IBC201

Figure 5-13. 1.0-bit Hardamard coding BER = 10^{-2}.



8761-18

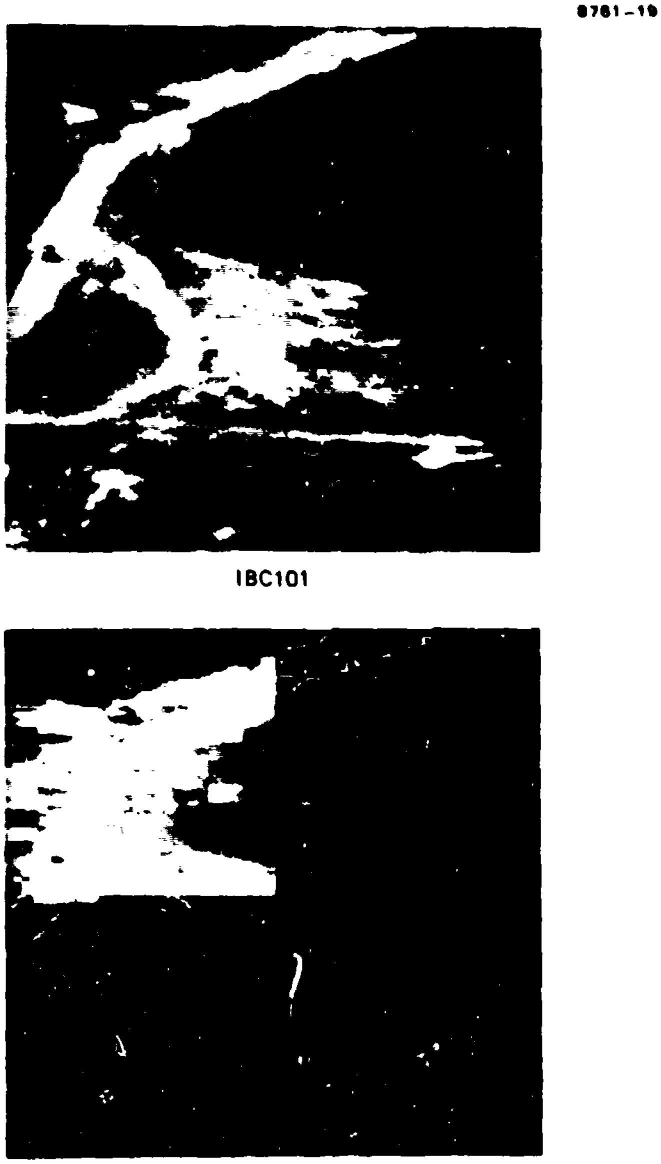
IBC101

IBC209

IBC107

IBC201

Figure 5-14. Alternate 1-bit Hadamard coding. BER = 10^{-2}.

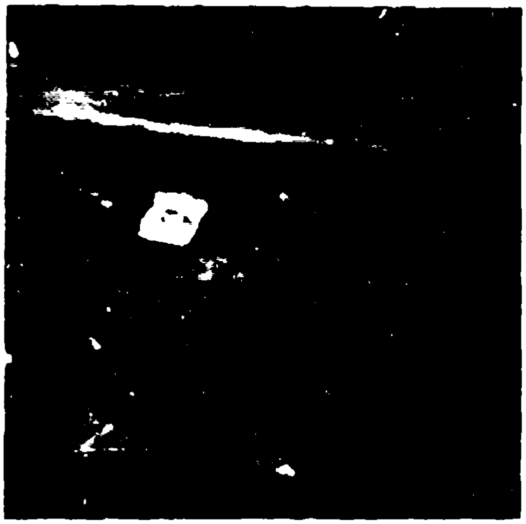


IBC101

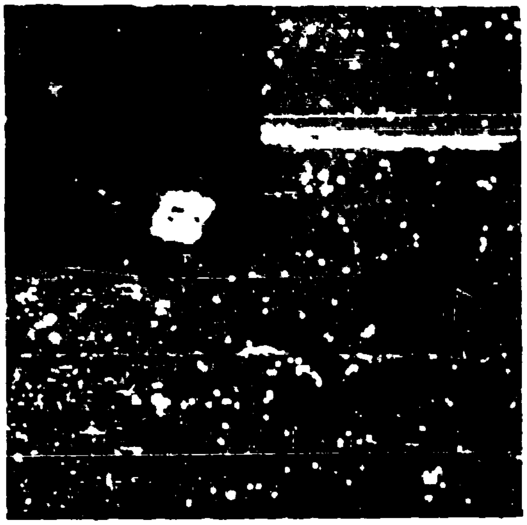
1000:1 BANDWIDTH COMPRESSION
HADAMARD CODING

Figure 5-15. IBC reconstructed imagery, data rate <50 kbs. BER = 10^{-2}.

8781-21



IBC107



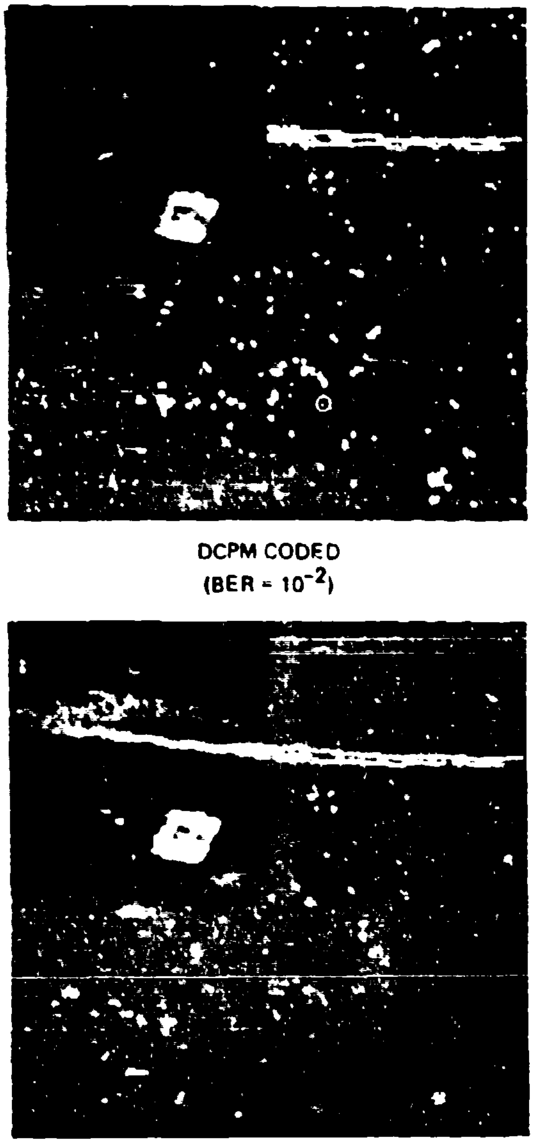
1000:1 BANDWIDTH COMPRESSION
DPCM CODED, BER = 10^{-2}

Figure 5-16. IBC reconstructed imagery, data rate <50 kbs.

orientation window was coded at 1 bpp and the edge map was transmitted at $(1/2)^2$ resolution. In Figure 5-16, the subimages were coded using DPCM at a BER = 10^{-2}, while in Figure 5-17 a comparison is made with Hadamard transform coding. Note that sharp edges (e.g., the road) in the orientation window are much better preserved with the transform coding.

Finally, the insertion of high-resolution target windows within a reduced-resolution orientation window did not provide a clear transition between target and background. Therefore, it was difficult to immediately focus on the targets. In order to provide a cue to the primary and secondary windows in the reconstructed image, a white border was placed around the primary target window, and a black border was placed around the secondary windows. This scheme proved to be quite appealing in providing a good balance between overall scene content presentation and the necessary cues for directing attention to the priority targets in the image. An example of the overall IBC reconstruction comparison is shown in Figure 5-18.

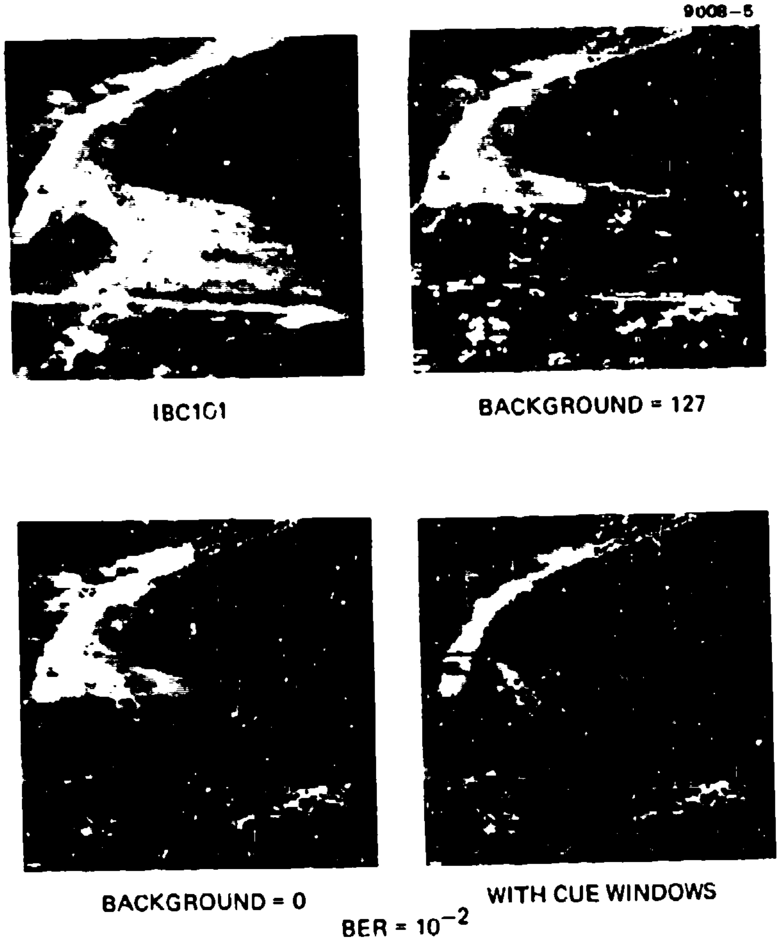
8761-22



DCPM CODED
(BER = 10^{-2})

HADAMARD CODED
(BER = 10^{-2})

Figure 5-17.
IBC reconstructed imagery comparison.



IBC101

BACKGROUND = 127

BACKGROUND = 0

WITH CUE WINDOWS

BER = 10^{-2}

Figure 5-18. IBC reconstruction comparison.

SECTION 6

CONCLUSIONS AND RECOMMENDATIONS

The results of the Intelligent Bandwidth Compression program have shown that a 1,000:1 bandwidth compression ratio for transmission of images is feasible in RPV applications. Realization of such a high compression ratio depends critically on a new approach toward image compression: the preservation of scene content information rather than global image fidelity. In this approach, advanced scene analysis techniques, such as automatic target cueing and image feature extraction, have been used extensively to derive scene content information. In addition, a rule-based control system has been developed and incorporated into the IBC system to direct the system operations.

The reconstructed images, which consists of a combination of edge map and full-resolution target windows updated at various frame rates, are easily interpreted. The edge map depicts general information on scene background, while the full-resolution target windows provide a detailed grey-scale picture of the detected targets. The orientation window, centered on the primary target, did not provide any additional clarity to the 1,000:1 compressed image. Often, it was a source of distraction in focusing on the targets. The bandwidth allocated for the orientation window can be better utilized for other resources, as will be proposed later.

We have found that the combination of a knowledge-base system and Pascal language controller provided the flexibility necessary to implement the complexities of the IBC system simulation. System modifications were carried out without major reconfiguration. This flexibility was very useful in the development of the IBC system.

The model-based classifier incorporated into the auto-cuer provided a second level recognition capability for classes of targets which were segmented into multiple pieces. By using the feature characteristics of each segment in conjunction with their relative dispositions, the class of trucks and jeeps were properly recognized by the classifier.

The video tape sequence of a simulation of the IBC system provided a real-time assessment of the reconstructed imagery. It was discovered at this time that motion compensation of the sensor platform was essential. Because of the different update rates for the edge map and primary target window (1 fps and 7-1/2 fps, respectively), the relative target position in the edge map varied wildly if the sensor line-of-sight was not very stable between the 1 fps updates. In a subsequent improvement of the video tape simulation of the IBC system, motion compensation was manually added. The resulting simulation was far superior to the one without motion compensation.

In addition, it was also discovered that the orientation window surrounding the primary target tended to distract from the target, rather than help it. Therefore, this allocation of the bandwidth can be better utilizes by reassigning it to update the secondary targets at 3 fps. This was also included in the second video tape simulation. A revised resource allocation which takes this into account is shown in Table 6-1.

Finally, from the results of the IBC system study, it appears that greater than 1,000:1 compression ratio for image transmission can be realistically achieved, and can provide reconstructed images which are

Table 6-1. IBC Resource Allocation

9259-6

DATA COMPOSITION	SIZE	FPS	BPP	DATA RATE kbit/sec
EDGE MAP	256 x 256[a] PIXELS	1.0	1.0	16.4
PRIORITY TARGET WINDOW	32 x 32 PIXELS	7.5	1.5	11.5
SECONDARY TARGET WINDOWS (6)	32 x 32 PIXELS	3.0	1.5	27.7
SYMBOLIC DESCRIPTORS	5 x 200 CHAR	1.0	7.0	7.0
			TOTAL	62.6

[a] EDGE MAP TRANSMITTED AT REDUCED RESOLUTION (128 x 128 PIXEL) AND REPLICATED TO 256 x 256 SIZE AT GROUND STATION.

useful for RPV applications. The degree of further compression depends critically on more reliable scene analysis, second-level semantics, and appropriate chain coding of the edge map for further bandwidth reduction. The usefulness of the reconstructed images can only be assessed fully by an extensive human factors study of lengthy simulated results of the reconstructed imagery.

REFERENCES

1. "Demonstration of Westinghouse Automatic Cueing Techniques Using NVL Imagery," Contract DAAG53-75-C-0025 to Night Vision Laboratory.

2. "FLIR Image Analysis with the Autoscreener Computer Simulation," Vols. 1 and 2, Contract DAAG53-75-C-0246 to Night Vision Laboratory.

3. "A Discussion of Design Goals and Hardware Implementation for an Automatic Target Cueing System," Quarterly Report (July 30, 1976) on Contract DAAG53-76-C-0138, Westinghouse Defense and Electronic Systems Center.

5. D. Milgram, "Region Extraction Using Convergent Evidence," Proc.: Image Understanding Workshop, April 1977, Science Applications, Inc.

6. D. Panda, "Segmentation of FLIR Images Pixel Classification," Proc.: Image Understanding Workshop, April 1977, Science Applications, Inc.

7. M. Geokezas, R. Jennewine, and G.P. Swanland, "A Real Time Imagery Screener," Report on work conducted on USAF Contract AFAL-TR-74-184, October 1974.

8. "Processing of FLIR Data on DICIFER," Contract DAAG53-75-C-0277 for Night Vision Laboratory, by Pattern ANalysis and Recognition Corp.

9. M. Geokezas and R. Jennewine, "Target Screener/FLIR System," Proceedings of the SPIE,Vol. 101, pp. 84-90, April 1977.

10. G.E. Tisdale, "A Digital Image Processor for Authomatic Target Cueing, Navigation, and Change Detection," Proceedings of the SPIE, Vol. 101, pp. 112-119, April 1977.

11. "A Discussion of Hardware Implementation and Fabrication for an Automatic Target Cueing System," Quarterly Report (Jan. 31, 1977) on Contract DAAG53-76-C-0138.

12. G.R. Nudd, P.A. Hygaard, and J.L. Erickson, "Image Processing Techniques Using Charge-Transfer Devices," Proc.: Image Understanding Workshop, October 1977, Science Applications, Inc.

13. B. Bullock, "Real World Scene Analysis in Perspective," Proc. 1975 Annual ACM Conf., Oct. 1975.

14. B. Bullock, "Unstructured Control Concepts in Scene Analysis," Proc. 8tth Annual Southeast Symposium on Sept. Theory," 1976.

15. B. Bullock, "The Necessity for a Theory of Specialized Vision," Computer Vision Systems, E. Riseman (Ed.,), Academic Press, 1979.

16. D. Waterman, "Pattern-Directed Inference Systems, Academic Press, 1979.

17. D.J. Sakrison, H. Halter, and M. Mostafavi, "Properties of the Human Visual System as Related to the Encoding of Images," In New Directions in Signal Processing in Communications and Control, NATO Advanced Studies Institute Series, Nordhoff International Publishing Company, 1974.

18. L. Fung and K.S. Fu, "An Error-Correcting Syntactic Decoder for Computer Networks," Int. Journal of Computer & Information Science, Vol. 5, No. 1, 1976.